Statistical Relational Learning Notes

Ganesh Ramakrishnan

October 8, 2008

Contents

1 Graphical Models **3**

1.1 Semantics of Graphical Models 4

1.1.1 Directed Graphical Models 4

1.1.2 Undirected Graphical Models 10

1.1.3 Comparison between directed and undirected graphical models 13

1.2 Inference 15

1.2.1 Elimination Algorithm 16

1.2.2 Sum-product Algorithm 18

1.2.3 Max Product Algorithm 22

1.2.4 Junction Tree Algorithm 27

1.2.5 Junction Tree Propagation 30

1.2.6 Constructing Junction Trees 31

1.2.7 Approximate Inference using Sampling 35

1.3 Factor Graphs 45

1.4 Exponential Family 47

1.5 A Look at Modeling Continuous Random Variables 49

1.6 Exponential Family and Graphical Models 53

1.6.1 Exponential Models and Maximum Entropy 55

1.7 Model fitting 59

1.7.1 Sufficient Statistics 60

1.7.2 Maximum Likelihood 62

1.7.3 What the Bayesians Do 64

1.7.4 Model Fitting for Exponential Family 66

1.7.5 Maximum Likelihood for Graphical Models 68

1.7.6 Iterative Algorithms 70

1.7.7 Maximum Entropy Revisted 73

1.8 Learning with Incomplete Observations 75

1.8.1 Parameter Estimation for Mixture Models 76

1.8.2 Expectation Maximization 78

1.9 Variational Methods for Inference 82

2 Linear Algebra **89**
2.1 Linear Equations . . . 89
2.2 Vectors and Matrices . . . 91
2.3 Solution of Linear Equations by Elimination . . . 95
2.3.1 Elimination as Matrix Multiplication . . . 97
2.4 Solution of Linear Equations by Matrix Inversion . . . 102
2.4.1 Inverse Matrices . . . 102
2.4.2 Gauss-Jordan Elimination . . . 104
2.5 Solution of Linear Equations using Vector Spaces . . . 106
2.5.1 Vector Spaces and Matrices . . . 106
2.5.2 Null Space . . . 108
2.6 Elimination for Computing the Null Space (Ax = 0) . . . 109
2.6.1 Pivot Variables and Row Echelon Form . . . 109
2.6.2 Reduced Row Echelon Form . . . 112
2.6.3 Solving $A\mathbf{x} = \mathbf{b}$ . . . 114
2.7 Independence, Basis, and Dimension . . . 119
2.7.1 Independence . . . 120
2.7.2 Basis and Dimension . . . 121
2.8 Matrix Spaces . . . 124
2.9 Orthogonality and Projection . . . 126
2.9.1 Projection Matrices . . . 127
2.9.2 Least Squares . . . 127
2.9.3 Orthonormal Vectors . . . 129
2.9.4 Gram-Schmidt Orthonormalization . . . 131
2.9.5 Fourier and Wavelet Basis . . . 132
2.10 Determinants . . . 133
2.10.1 Formula for determinant . . . 138
2.10.2 Formula for Inverse . . . 140
2.11 Eigenvalues and Eigenvectors . . . 141
2.11.1 Solving for Eigenvalues . . . 142
2.11.2 Some Properties of Eigenvalues and Eigenvectors . . . 143
2.11.3 Matrix Factorization using Eigenvectors . . . 148
2.12 Positive Definite Matrices . . . 149
2.12.1 Equivalent Conditions . . . 150
2.12.2 Some properties . . . 151
2.13 Singular Value Decomposition . . . 152
2.13.1 Pseudoinverse . . . 154

3 Convex Optimization **157**
3.1 Introduction . . . 157
3.1.1 Mathematical Optimization . . . 157
3.1.2 Some Topological Concepts in $\Re^n$ . . . 158
3.1.3 Optimization Principles for Univariate Functions . . . 160
3.1.4 Optimization Principles for Multivariate Functions . . . 175
3.1.5 Absolute extrema and Convexity . . . 196
3.2 Convex Optimization Problem . . . 197

3.2.1 Why Convex Optimization? 197
3.2.2 History 198
3.2.3 Affine Set 199
3.2.4 Convex Set 199
3.2.5 Examples of Convex Sets 201
3.2.6 Convexity preserving operations 204
3.2.7 Convex Functions 209
3.2.8 Convexity and Global Minimum 210
3.2.9 Properties of Convex Functions 212
3.2.10 Convexity Preserving Operations on Functions 223
3.3 Convex Optimization Problem 224
3.4 Duality Theory 226
3.4.1 Lagrange Multipliers 226
3.4.2 The Dual Theory for Constrained Optimization 232
3.4.3 Geometry of the Dual 236
3.4.4 Complementary slackness and KKT Conditions 239
3.5 Algorithms for Unconstrained Minimization 243
3.5.1 Descent Methods 244
3.5.2 Newton's Method 249
3.5.3 Variants of Newton's Method 253
3.5.4 Gauss Newton Approximation 253
3.5.5 Levenberg-Marquardt 255
3.5.6 BFGS 255
3.5.7 Solving Systems Large Sparse Systems 257
3.5.8 Conjugate Gradient 265
3.6 Algorithms for Constrained Minimization 268
3.6.1 Equality Constrained Minimization 268
3.6.2 Inequality Constrained Minimization 271
3.7 Linear Programming 274
3.7.1 Simplex Method 277
3.7.2 Interior point barrier method 284
3.8 Least Squares 288
3.8.1 Linear Least Squares 289
3.8.2 Least Squares with Linear Constraints 294
3.8.3 Least Squares with Quadratic Constraints 297
3.8.4 Total Least Squares 298

Chapter 1

Graphical Models

Graphical models [Lau96, CGH97, Pea88, Jor98, Jen01] provide a pictorial representation of the way a joint probability distribution factorizes into a product of factors. They have been widely used in the area of computer vision, control theory, bioinformatics, communication, signal processing, sensor networks, neurovision, *etc.*

There is a relation between the conditional independence properties of a joint distribution and its factorization properties. Each of these properties can be represented graphically. This allows us to

1. organize complicated mathematics through graph based algorithms for calculation and computation and thus save complicated computations (in many cases, map the pictorial representation onto computations directly)

2. gain new insights into existing models; it is a standard practice to modify an existing model by addition/delection of nodes or links and adopt it to the problem at hand

3. motivate new models by simply ammending pictures

However, everything that you could do in graphical models using pictures could also be done without pictures, by grinding through all the mathematics while consistently using the innocuous-looking sum (1.1) and product (1.2) rules of probability

$$\Pr(X) = \sum_{y \in \mathcal{Y}} \Pr(X = x, Y = y) \tag{1.1}$$

$$\Pr(X = x, Y = y) = \Pr(X = x \mid Y = y) \Pr(Y = y) \tag{1.2}$$

where X and Y are discrete random variables, assuming values $x \in \mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ and $y \in \mathcal{Y} = \{y_1, y_2, \ldots, y_n\}$ respectively. A combination of the two rules yields the Bayes theorem:

$$\begin{aligned} \Pr(X = x_i, Y = y_j) &= \frac{\Pr(X = x_i \mid Y = y_j)\Pr(Y = y_j)}{\Pr(X = x_i)} \\ &= \frac{\Pr(X = x_i \mid Y = y_j)\Pr(Y = y_j)}{\sum\limits_{y_j \in \mathcal{Y}} \Pr(X = x_i \mid Y = y_j)\Pr(Y = y_j)} \end{aligned} \tag{1.3}$$

The two main kinds of graphical models are *directed* and *undirected* models. The problems we will address in graphical models include

1. *Inference:* Broadly, there are two inference techniques for graphical models, *viz.*, exact and approximate inference. Exact inference is appropriate if the graphic is a tree, since it is a linear time algorithm. But for complex graphical models, exact inference may or may not be appropriate, since exact algorithms could be very slow. In such cases, approximate inference schemes are often resorted to. Markov chain monte carlo (which is exact if there were an infinite amount of computing resources and approximate otherwise) and variational inference (by approximating the analytical form for the posterior distribution) are two popular techniques. While variational techniques scale better, their other strengths and weaknesses are complementary to those of MCMC. An often adopted stepping stone for explaining variational inference is the expectation maximization algorithm (EM) and we will take the same route.

2. *Learning:*

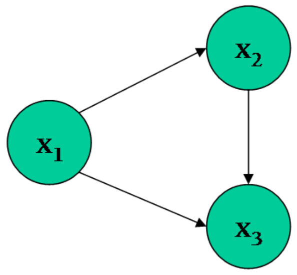
1.1 Semantics of Graphical Models

We will first discuss the semantics of graphical models, both directed and undirected. In the sections that follow, we will discuss the computational aspects of graphical models - in particular, inferencing and learning techniques.

1.1.1 Directed Graphical Models

We will start with the example of a directed graphical model. Consider an arbitrary joint distribution $\Pr(X_1 = x_1, X_2 = x_2, X_3 = x_3)$ over three discrete random variables X_1, X_2 and X_3 that assume values $x_1 \in \mathcal{X}_1$, $x_2 \in \mathcal{X}_2$ and $x_3 \in \mathcal{X}_3$ respectively. Denoting[1] $\Pr(X_i = x_i)$ by $p(x_i)$ and $\Pr(X_1 = x_1, X_2 = x_2, X_3 = x_3)$ by $p(x_1, x_2, x_3)$ and applying the product rule of probability successively, we obtain

[1] As a convention, we will use capital letters to denote random variables and lower case letters to denote their realizations.



$$p(x_1, x_2, x_3) = p(x_1)p(x_2, x_3 \mid x_1) = p(x_1)p(x_2 \mid x_1)p(x_3 \mid x_1, x_2) \tag{1.4}$$

The successive application of the product rule is often termed as the *chain rule*. We should note that this rule applies even if x_1, x_2 and x_3 happen to be continuous random variables (in which case p is a density function) or vectors of random variables. We should note that this factorization is quite non-symmetrical in the three random variables. This factorization can be represented in the form of the following directed graph: There is one node for each variable. We draw a directed edge between every conditioning variable (*i.e.* the corresponding node) to the conditioned variable. The way to go from a graph to the factorization of the joint distribution is to write down the product of the conditional distribution of every node (*i.e.*, the corresponding variable) conditioned on its parents within the graph. In the example above, x_1 had no parent, and therefore the term corresponds to its conditional $p(x_1)$ turned out to be its marginal distribution.

The factorization in the last example holds for any joint distribution over any three variables and the graph is therefore uninformative. In fact, any completely connected graph will be uninformative, as we will soon see. What interests us in graphical models is not the *presence* of edges but rather, the *absence* of edges. Since the graph in the previous example had no missing edges, it was uninteresting.

Definition 1 *Let $\mathcal{R} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables, with each X_i ($1 \leq i \leq n$) assuming values $x_i \in \mathcal{X}_i$. Let $\mathcal{X}_S = \{X_i \mid i \in S\}$ where $S \subseteq \{1, 2, \ldots, n\}$. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a directed acyclic graph with vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ such that each edge $e = (i, j) \in \mathcal{E}$ is a directed edge. We will assume a one to one correspondence between the set of variables $\mathcal{R}$ and the vertex set $\mathcal{V}$; vertex i will correspond to random variable X_i. Let π_i be the set of vertices from which there is edge incident on vertex i. That is, $\pi_i = \{j \mid j \in \mathcal{V},\ (j, i) \in \mathcal{E}\}$. Then, the family $\mathcal{F}(\mathcal{G})$ of joint distributions associated with the DAG[2] $\mathcal{G}$ is specified by the factorization induced by $\mathcal{G}$ as follows:*

[2] As we saw earlier, the family of probability distributions specified by the related formalism of undirected graphical models is somewhat different.

$$\mathcal{F}(\mathcal{G}) = \left\{ p(\mathbf{x}) \middle| p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{x}_{\pi_i}),\ p(x_i \mid \mathbf{x}_{\pi_i}) \forall\ 1 \le i \le n \ge 0,\ \sum_{x_i \in \mathcal{X}_i} p(x_i \mid \mathbf{x}_{\pi_i}) = 1 \right\} \tag{1.5}$$

where, $\mathbf{x}$ *denotes the vector of values* $[x_1, x_2, \ldots, x_n]$ *and* x_i *is the value assumed by random variable* X_i *and* $\mathbf{x}_{\pi_i}$ *denotes the vector of values from* $\mathbf{x}$*, composed from positions in* π_i*.*

For notational convenience with directed acyclic graphs, it is a common practice to assume a topological ordering on the indices of vertices in $\mathcal{V}$ so that $\pi_i \subseteq \mu_{i-1} = \{1, 2, \ldots, i-1\}$. Note that, by the chain rule, the following factorization always holds:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i \mid \mathbf{x}_{\mu_{i-1}}) \tag{1.6}$$

Making use of the sum rule, in conjunction with (1.5), for any $p \in \mathcal{F}(\mathcal{G})$, we have

$$\begin{aligned} p(\mathbf{x}_{\mu_i}) &= \sum_{x_{i+1} \in \mathcal{X}_{i+1}, \ldots, x_n \in \mathcal{X}_n} p(\mathbf{x}) \\ &= \sum_{x_{i+1} \in \mathcal{X}_{i+1}, \ldots, x_n \in \mathcal{X}_n} p(x_1) p(x_2 \mid \mathbf{x}_{\pi_2}) \ldots p(x_i \mid \mathbf{x}_{\pi_i}) \end{aligned} \tag{1.7}$$

Since the vertex indices are topologically ordered, it can be proved using the principle of induction (working backwards from n) on the basis of the sum rule in (1.7), that for any $p \in \mathcal{F}(\mathcal{G})$:

$$p(\mathbf{x}_{\mu_i}) = \prod_{j=1}^{i-1} p(x_i \mid \mathbf{x}_{\pi_i}) \tag{1.8}$$

Contrasting (1.6) against (1.5), we can think of the set of probability distribution $\mathcal{F}(\mathcal{G})$ as a sort of restricted class of distributions that arises from throwing away some of the dependencies. In particular, if $p \in \mathcal{F}(\mathcal{G})$ then

$$p(x_i \mid \mathbf{x}_{\mu_{i-1}}) = p(x_i \mid \mathbf{x}_{\pi_i})$$

that is, X_i is independent of $\mathbf{X}_{\mu_{i-1}}$, given $\mathbf{X}_{\pi_i}$. The independence is denoted by: $X_i \perp \mathbf{X}_{\mu_{i-1}} \mid \mathbf{X}_{\pi_i}$. This leads us to another approach to defining the class of probability distributions based on a DAG $\mathcal{G}$.

Definition 2 *Let $\mathcal{R} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables, with each X_i ($1 \leq i \leq n$) assuming values $x_i \in \mathcal{X}_i$. Let $\mathbf{X}_\mathcal{S} = \{X_i \mid i \in \mathcal{S}\}$ where $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a directed acyclic graph with vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ such that each edge $e = (i, j)$ is a directed edge. Let $\pi_i = \{j \mid j \in \mathcal{V},\ (j, i) \in \mathcal{E}\}$. Then, the family $\mathcal{C}(\mathcal{G})$ of joint distributions associated with the DAG $\mathcal{G}$ is specified by the conditional independence induced by $\mathcal{G}$ as follows:*

$$\mathcal{C}(\mathcal{G}) = \left\{ p(\mathbf{x}) \middle| X_i \perp \mathbf{X}_{\mu_{i-1}} \mid \mathbf{X}_{\pi_i}\ \forall\ 1 \leq i \leq n,\ \sum_{\mathbf{x}} p(\mathbf{x}) = 1 \right\} \tag{1.9}$$

We will next show that the class $\mathcal{F}(\mathcal{G})$ defined in terms of factorizations is equivalent to the class $\mathcal{C}(\mathcal{G})$ defined in terms of independences. This is called the Hammersley Clifford theorem.

Theorem 1 *The sets $\mathcal{F}(\mathcal{G})$ and $\mathcal{C}(\mathcal{G})$ are equal. That is $p \in \mathcal{F}(\mathcal{G})$ iff $p \in \mathcal{C}(\mathcal{G})$*

Proof: $\Leftarrow$: We will first prove that $\mathcal{F}(\mathcal{G}) \subseteq \mathcal{C}(\mathcal{G})$. Let $p \in \mathcal{F}(\mathcal{G})$. We will prove that $p \in \mathcal{C}(\mathcal{G})$, that is, $p(x_i \mid \mathbf{x}_{\mu_{i-1}}, \mathbf{x}_{\pi_i}) = p(x_i \mid \mathbf{x}_{\pi_i})$. This trivially holds for $i = 1$, since $\mathbf{x}_{\pi_i} = \emptyset$. For $i = 2$:

$$p(x_1, x_2) = p(x_1)p(x_1 \mid x_2) = p(x_1)p(x_1 \mid \mathbf{x}_{\pi_2})$$

where, the first equality follows by chain rule, whereas the second equality follows by virtue of (1.8). Consequently,

$$p(x_1 \mid x_2) = p(x_1 \mid \mathbf{x}_{\pi_2})$$

Assume that $p(x_i \mid \mathbf{x}_{\mu_{i-1}}) = p(x_i \mid \mathbf{x}_{\pi_i})$ for $i \leq k$. For $i = k+1$, it follows from chain rule and from (1.8) that

$$p(\mathbf{x}_{\mu_{k+1}}) = \prod_{i=1}^{k+1} p(x_i \mid \mathbf{x}_{\mu_{i-1}}) = \prod_{i=1}^{k+1} p(x_i \mid \mathbf{x}_{\pi_i})$$

Making use of the induction assumption for $i \leq k$ in the equation above, we can derive that

$$p(x_k \mid \mathbf{x}_{\mu_{k-1}}) = p(x_k \mid \mathbf{x}_{\pi_k})$$

By induction on i, we obtain that $p(x_i \mid \mathbf{x}_{\mu_{i-1}}) = p(x_k \mid \mathbf{x}_{\pi_i})$ for all i. That is, $p \in \mathcal{C}(\mathcal{G})$. Since this holds for any $p \in \mathcal{F}(\mathcal{G})$, we must have that $\mathcal{F}(\mathcal{G}) \subseteq \mathcal{C}(\mathcal{G})$.

$\Rightarrow$: Next we prove that $\mathcal{C}(\mathcal{G}) \subseteq \mathcal{F}(\mathcal{G})$. Let $p' \in \mathcal{C}(\mathcal{G})$ satisfy the conditional independence assertions. That is, for any $1 \leq i \leq n$, $p'(x_i \mid \mathbf{x}_{\mu_{i-1}}) = p'(x_i \mid \mathbf{x}_{\pi_i})$. Then by chain rule, we must have:

$$p'(\mathbf{x}_{\mu_n}) = \prod_{i=1}^{n} p'(x_i \mid \mathbf{x}_{\mu_{i-1}}) = \prod_{i=1}^{k+1} p'(x_i \mid \mathbf{x}_{\pi_i})$$

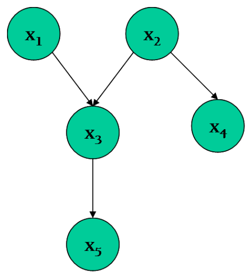


Figure 1.1: A directed graphical model.

which proves that $p' \in \mathcal{F}(\mathcal{G})$ and consequently that $\mathcal{C}(\mathcal{G}) \subseteq \mathcal{F}(\mathcal{G})$ $\Box$

As an example, we will discuss the directed graphical model, as shown in Figure 1.1. Based on theorem 1, the following conclusions can be drawn from the graphical representation of a family of distributions represented by Figure 1.1.

1. Given the value of X_3, the values of X_1, X_2 and X_4 will be completely uninformative about the value of X_5. That is, $(X_5 \perp \{X_1, X_2, X_4\} \mid \{X_3\})$. Similarly, given the value of X_2, the values of X_1 and X_3 will be completely uninformative about the value of X_4. That is, $(X_4 \perp \{X_1, X_3\} \mid \{X_2\})$.

2. Secondly, since $p \in \mathcal{F}(\mathcal{G})$, we have

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_1)p(x_2)p(x_3 \mid x_1, x_2)p(x_4 \mid x_2)p(x_5|x_3)$$

What about other independence assertions? Is X_5 independent of X_4 given X_2? Is X_3 independent of X_4, given X_2? The answer to both these questions happens to be yes. And these could be derived using either of the equivalent definitions of graphical models. In fact, some such additional conditional independence assertions can always follow from the two equivalent definitions of graphical models. Before delving into these properties, we will define an important concept called d-separation, introduced by Pearl [Pea88].

Definition 3 *A set of nodes $\mathcal{A}$ in a directed acyclic graph $\mathcal{G}$ is d-separated from a set of nodes $\mathcal{B}$ by a set of nodes $\mathcal{C}$, iff* **every** *undirected path from a vertex $A \in \mathcal{A}$ to a vertex $B \in \mathcal{B}$ is 'blocked'. An undirected path between A and B is blocked by a node C either (i) if $C \in \mathcal{C}$ and both the edges (which might be the same) on the path through C are directed away from C (C is then called a tail-to-tail node) or (ii) if $C \in \mathcal{C}$ and of the two edges (which might be the same) on the path through C, one is directed toward C while the other is directed away from C (C is then called a head-to-tail node) or (iii) if $C \notin \mathcal{C}$ and both the edges (which might be the same) on the path through C are directed toward C (C is then called a head-to-head node).*

In Figure 1.1, node X_2 blocks the only path between X_3 and X_4, node X_3 blocks the path between X_2 and X_5. Whereas, node X_3 does not block the path between X_1 and X_2. Consequently, $\{X_3\}$ and $\{X_4\}$ are d-separated by $\{X_2\}$, while $\{X_2\}$ and $\{X_5\}$ are d-separated by $\{X_3\}$. However, $\{X_1\}$ and $\{X_2\}$ are not d-separated by $\{X_3\}$, since X_3 is a head-to-head node. X_3 does not d-separate X_1 and X_2 even though it separates them. Thus, not every pair of (graph) separated nodes in the graph need be d-separated. We next define a family of probability distribution that have independences characterized by d-separation.

Definition 4 *The set of probability distributions $\mathcal{D}(\mathcal{G})$ for a DAG $\mathcal{G}$ is defined as follows:*

$$\mathcal{D}(\mathcal{G}) = \{p(\mathbf{x})\,|\mathbf{X}_{\mathcal{A}} \perp \mathbf{X}_{\mathcal{B}} \mid \mathbf{X}_{\mathcal{C}},\ whenever\ \mathcal{A}\ and\ \mathcal{B}\ are\ d-separated\ by\ \mathcal{C}\,\} \tag{1.10}$$

It can be proved that the notion of conditional independence is equivalent to the notion of d-separation in DAGs. That is,

Theorem 2 *For any directed acyclic graph $\mathcal{G}$, $\mathcal{D}(\mathcal{G}) = \mathcal{C}(\mathcal{G}) = \mathcal{F}(\mathcal{G})$.*

Thus, in Figure 1.1. $\{X_3\} \perp \{X_4\} \mid \{X_2\}$ and $\{X_2\} \perp \{X_5\} \mid \{X_3\}$. Whereas, $\{X_1\} \not\perp \{X_2\} \mid \{X_3\}$. We could think of X_1 and X_2 as completing explanations for X_3. Thus, given a value of X_3, any value of X_1 will, to some extent 'explain away' the value of X_3, thus withdrawing the independence of X_2. In terms of a real life example, if X_1, X_2 and X_3 are discrete random variables corresponding to 'the color of light', 'the surface color' and 'the image color' respectively, then, given the value of X_3 (image color), any value of X_1 (color of light) will explain away the color of the image, thus constraining the values that X_3 (surface color) might take. On the other hand, $\{X_1\} \perp \{X_2\} \mid \{\}$. What about the independence of X_1 and X_2 given X_5? The path X_1, X_3, X_5, X_3, X_2 involves a head-to-head node X_5 and therefore, $X_1 \not\perp X_2 \mid \{X_5\}$. The Bayes ball algorithm provides a convenient algorithmic way for deciding if $\mathbf{X}_{\mathcal{A}} \perp \mathbf{X}_{\mathcal{B}} \mid \mathbf{X}_{\mathcal{C}}$, by using the d-separation property.

Bayesian Networks and Logic

The logical component of Bayesian networks essentially corresponds to a propositional logic program. This has already been observed by Haddawy [1994] and Langley [1995]. Langley, for instance, does not represent Bayesian networks graphically but rather uses the notation of propositional definite clause programs. Consider the following program. This program encodes the structure of the blood type Bayesian network in Figure 1.2. Observe that the random variables in this notation correspond to logical atoms. Furthermore, the direct influence relation in the Bayesian network corresponds to the immediate consequence operator.

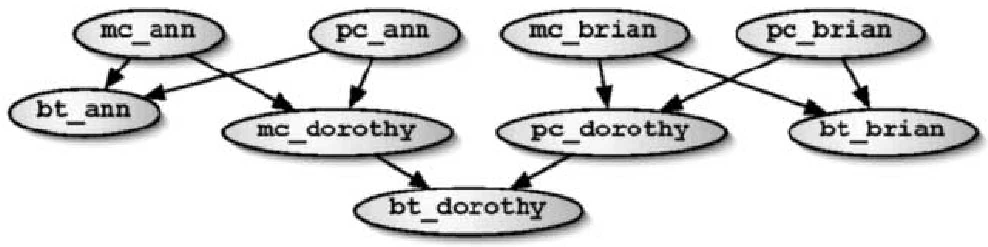


Figure 1.2: The graphical structure of a Bayesian network modeling the inheritance of blood types within a particular family.

$PC(ann).$	$PC(brian).$
$MC(ann).$	$MC(brian).$
$MC(dorothy) :- MC(ann), PC(ann).$	$PC(dorothy) :- MC(brian), PC(brian).$
$BT(ann) :- MC(ann), PC(ann).$	$BT(brian) :- MC(brian), PC(brian).$
$BT(dorothy) :- MC(dorothy), PC(dorothy).$	

1.1.2 Undirected Graphical Models

We will move on to an undirected graphical models (also known as Markov Random fields) primer, while drawing parallels with the directed counterpart. An undirected graph $\mathcal{G}$ is a tuple $< \mathcal{V}, \mathcal{E} >$ where $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and such that each edge $e = (i, j) \in \mathcal{E}$ is a directed edge. While the conditional independence property for directed graphical models was tricky (involving concepts such as d-separation, *etc.*), the conditional independence property for undirected models is easier to state. On the other hand, the factorization property for directed graphical models simply involved local conditional probabilities as factors. It is however not as simple with undirected graphical models. Taking the easier route, we will first define the conditional independence property for undirected graphical models. Toward that, we introduce the notion of graph separation.

Definition 5 *Given an undirected graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, and $\mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathcal{V}$, we say that $\mathcal{C}$ separates $\mathcal{A}$ from $\mathcal{B}$ in $\mathcal{G}$ if every path from any node $A \in \mathcal{A}$ to any node $B \in \mathcal{B}$ passes through some node $C \in \mathcal{C}$. C is also called a separator or a vertex cut set in $\mathcal{G}$.*

In Figure 1.3, the set $\mathcal{A} = \{X_1, X_7, X_8\}$ is separated from $\mathcal{B} = \{X_3, X_4, X_5\}$ by the (vertex cut) set $\mathcal{C} = \{X_2, X_6\}$. It is easy to see that separation is symmetric in $\mathcal{A}$ and $\mathcal{B}$. This simple notion of separation gives rise to a conditional indpendence assertion for undirected graphs. A random vector $\mathbf{X}_{\mathcal{C}}$ for $\mathcal{C} \subseteq \mathcal{V}$

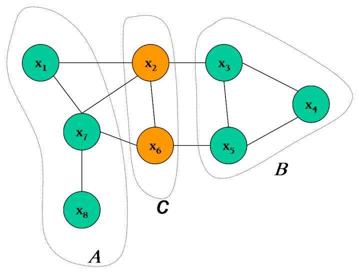


Figure 1.3: A directed graphical model.

is said to be *markov* with respect to a graph $\mathcal{G}$ if $\mathbf{X}_{\mathcal{A}} \perp \mathbf{X}_{\mathcal{B}} \mid \mathbf{X}_{\mathcal{C}}$ whenever $\mathcal{C}$ separates $\mathcal{A}$ from $\mathcal{B}$. That is, the random variables corresponding to the vertex cut set acts like a mediator between the values assumed by variables in $\mathcal{A}$ and $\mathcal{B}$ so that the variables in $\mathcal{A}$ and $\mathcal{B}$ are independent of each other, if we knew the values of variables in $\mathcal{C}$. It is straightforward to develop a 'reachability' algorithm (as with the bayes ball algorithm) for undirected graphs, to assess conditional independence assumptions. Based on the definition of markov random vector, we next define the family $\mathcal{M}(\mathcal{G})$ of distributions associated with an undirected graph $\mathcal{G}$.

Definition 6 *Let $\mathcal{R} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables, with each X_i ($1 \leq i \leq n$) assuming values $x_i \in \mathcal{X}_i$. Let $\mathbf{X}_{\mathcal{S}} = \{X_i \mid i \in \mathcal{S}\}$ where $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be an undirected graph with vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ such that each edge $e = (i, j) \in \mathcal{E}$ is an undirected edge. Then, the family $\mathcal{M}(\mathcal{G})$ of joint distributions associated with $\mathcal{G}$ is specified as follows:*

$$\mathcal{M}(\mathcal{G}) = \{p(\mathbf{x})\,|\mathbf{X}_{\mathcal{A}} \perp \mathbf{X}_{\mathcal{B}} \mid \mathbf{X}_{\mathcal{C}}\ \forall\ \mathcal{A}, \mathcal{B}, \mathcal{C} \subseteq \mathcal{V},\ \textit{whenever } \mathcal{C} \textit{ seperates } \mathcal{A} \textit{ from } \mathcal{B}\} \tag{1.11}$$

As in the case of directed models, there is another family of probability distributions that could be specified for a given undirected graph $\mathcal{G}$ based on a factorization assertion. The main difference is that, while the factorization for DAGs was obtained in terms of local conditional probabilities or local marginals, it turns out that this factorization is not possible for a general undirected graph (specifically when it has a cycle). Instead there is another notion of 'local' for undirected graphs – there should be no function involving any two variables X_i and X_j where $(i, j) \notin \mathcal{E}$ (otherwise, such a term will not break further, prohibiting assertions about conditional independences). Instead, we will have a function $\phi_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}})$ for every clique $\mathcal{C} \subseteq \mathcal{V}$, since a clique is a subset of vertices that all 'talk to' one another. The most general version of factorization will be one for which there is a function corresponding to each *maximal clique*; all other factorizations involving factors over smaller cliques will be specialized versions. These functions will be referred to as *compatibility* or *potential* functions. A *potential* or *compatibility* function on a clique $\mathcal{C}$ is a non-negative real valued function $\phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})$ defined over all instantiations $\mathbf{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n$ of $\mathbf{X}$. Other than these restrictions, the potential function can be arbitrary.

Definition 7 *Let $\mathcal{R} = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables, with each X_i ($1 \le i \le n$) assuming values $x_i \in \mathcal{X}_i$. Let $\mathbf{X}_{\mathcal{S}} = \{X_i \mid i \in \mathcal{S}\}$ where $\mathcal{S} \subseteq \{1, 2, \ldots, n\}$. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be an undirected graph with vertices $\mathcal{V} = \{1, 2, \ldots, n\}$ and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ such that each edge $e = (i, j) \in \mathcal{E}$ is an undirected edge. Then, the family $\mathcal{M}(\mathcal{G})$ of joint distributions associated with $\mathcal{G}$ is specified as follows:*

$$\mathcal{F}(\mathcal{G}) = \left\{ p(\mathbf{x}) \middle| p(\mathbf{x}) = \frac{1}{Z} \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}),\ \textit{such that } \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \in \Re^+, \forall \mathcal{C} \in \Pi \right\} \tag{1.12}$$

where, Π is the set of all cliques in $\mathcal{G}$, $\mathbf{x}$ denotes the vector of values $[x_1, x_2, \ldots, x_n]$, $x_i \in \mathcal{X}_i$ is the value assumed by random variable X_i and where each $\phi_{\mathcal{C}}$ is a potential function defined over the clique $\mathcal{C} \subseteq \mathcal{V}$. Without loss of generality, we can assume that Π is the set of maximal cliques in $\mathcal{G}$. The normalization constant Z is called the partition function and is given by

$$Z = \sum_{\mathbf{x}_1 \in \mathcal{X}_1, \ldots, \mathbf{x}_n \in \mathcal{X}_n} \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \tag{1.13}$$

The potential functions are typically represented as tables – each row listing a unique assignment of values to the random variables in the clique and the corresponding potential. Thus, the value $\phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})$ can be obtained by a simple table lookup.

The form of the potential function can be chosen based on the particular application at hand. For instance, the clique potential can be decomposed into a product of potentials defined over each edge of the graph. When the domain of each random variable is the same, the form of the potential can be chosen to either encourage or discourage similar configurations (such as similar disease infection for patients who are related) at adjacent nodes. The potential function is often interpreted as an energy function in the modeling of crystals, protein folding, *etc.*, where a minimum energy configuration is desirable. Frequently, the energy function is assumed to have the form $\phi_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}) = \exp(-\theta_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}}))$, which leads to the factorization as an exponential form distribution $p(\mathbf{x}) = \exp(-\sum_{\mathcal{C} \in \Pi} \theta_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) - \log Z)$. The quantities $\theta_{\mathcal{C}}(\mathbf{X}_{\mathcal{C}})$ are called sufficient statistics.

From our discussion on the equivalence of directed and undirected trees in terms of conditional independencies, we may be tempted to conclude that the factors for the undirected tree can be the local conditional probabilities. This is easily established if we prove that for strictly positive distributions, the definition of an undirected graphical model in terms of conditional independences is equivalent to the definition in terms of factorization, that is, $\mathcal{M}(\mathcal{G}) = \mathcal{F}(\mathcal{G})$.

Theorem 3 *For strictly positive distributions, $\mathcal{M}(\mathcal{G}) = \mathcal{F}(\mathcal{G})$. That is, $\mathcal{M}(\mathcal{G}) \cap \mathcal{D}^+ = \mathcal{F}(\mathcal{G}) \cap \mathcal{D}^+$, where, $\mathcal{D}^+ = \{p(\mathbf{x}) \,|p(\mathbf{x}) > 0, \forall \mathbf{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \ldots \times \mathcal{X}_n\}$.*

Proof: The proof that $\mathcal{M}(\mathcal{G}) \subseteq \mathcal{F}(\mathcal{G})$ is a bit involved and requires Mobius inversion. On the other hand, that $\mathcal{F}(\mathcal{G}) \subseteq \mathcal{M}(\mathcal{G})$ can be shown as follows. For any given $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ that are separated by $\mathcal{C} \subseteq \mathcal{V}$, consider the partitions $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ of Π:

$$\mathcal{P}_1 = \{\mathcal{K} \,|\mathcal{K} \in \Pi and \mathcal{K} \subseteq \mathcal{A} \cap \mathcal{C} \text{ and } \mathcal{K} \nsubseteq \mathcal{C}\}$$

$$\mathcal{P}_2 = \{\mathcal{K} \,|\mathcal{K} \in \Pi and \mathcal{K} \subseteq \mathcal{B} \cap \mathcal{C} \text{ and } \mathcal{K} \nsubseteq \mathcal{C}\}$$

$$\mathcal{P}_3 = \{\mathcal{K} \,|\mathcal{K} \in \Pi and \mathcal{K} \subseteq \mathcal{C}\}$$

Now, $p(\mathbf{x})$ can be factorized into factors involving cliques in $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$. Consequently,

$$\frac{p(\mathbf{x}_\mathcal{A}, \mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{C})}{p(\mathbf{x}_\mathcal{B}, \mathbf{x}_\mathcal{C})} = \frac{\prod\limits_{\mathcal{K} \in \mathcal{P}_1} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K}) \prod\limits_{\mathcal{K} \in \mathcal{P}_2} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K}) \prod\limits_{\mathcal{K} \in \mathcal{P}_3} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K})}{\sum\limits_{\mathbf{x}_\mathcal{A}} \prod\limits_{\mathcal{K} \in \mathcal{P}_1} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K}) \prod\limits_{\mathcal{K} \in \mathcal{P}_2} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K}) \prod\limits_{\mathcal{K} \in \mathcal{P}_2} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K})} = \frac{\prod\limits_{\mathcal{K} \subseteq \mathcal{A} \cup \mathcal{C}} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K})}{\sum\limits_{\mathbf{x}_\mathcal{A}} \prod\limits_{\mathcal{K} \subseteq \mathcal{A} \cup \mathcal{C}} \phi_\mathcal{K}(\mathbf{x}_\mathcal{K})} = p(\mathbf{x}_\mathcal{A} \mid \mathbf{x}_\mathcal{C})$$

□

While conditional independence is useful for modeling purposes, factorization is more useful for computatinal purposes.

1.1.3 Comparison between directed and undirected graphical models

Is there any difference between the undirected and directed formalisms? Are they equally powerful? Or is more powerful than the other. It turns out that there are families of probability distributions which can be represented using undirected models, whereas they have no directed counterparts. Figure 1.4 shows one such example. Imagine that random variables X_1 and X_3 represent hubs in a computer network, while X_2 and X_4 represent computers in the network. Computers do not interact directly, but only through hubs. Similarly, hubs interact only through computers. This leads to two independences: (i) conditioned on X_1 and X_3 (the hubs), nodes X_2 and X_4 (the computers) become independent and (ii) conditioned on X_2 and X_4, nodes X_1 and X_3 become independent. However, with a directed acyclic graph on four nodes, we will always have some head-to-head node and therefore, it is impossible to simultanesouly satisfy both conditions (i) and (ii) using a DAG. Larger bipartitie graphs

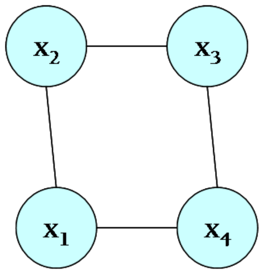


Figure 1.4: An undirected graphical model which has no equivalent directed model.

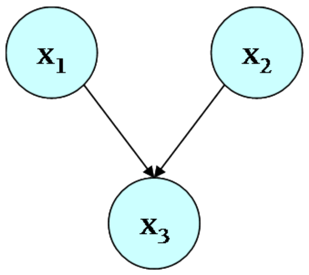


Figure 1.5: A directed graphical model which has no equivalent undirected model.

will have similar conditional independence assertions, which are inexpressible through DAGs.

Similarly, there are familiies of probability distibutions which can be represented using directed models, whereas they have no undirected counterparts. Figure 1.5 shows one such example and corresponds to the 'explaining away' phenomenon, which we discussed earlier in this chapter. The node X_3 is blocked if not observed (so that $\{X_1\} \perp \{X_2\} \mid \emptyset$), whereas it is unblocked if its value is known (so that $\{X_1\} \not\perp \{X_2\} \mid \{X_3\}$). Can you get this behaviour with an undirected graph? The answer is no. This is because, with an undirected graph, there is no way of getting dependence between X_1 and X_2 if they were apriori independent.

On the other hand, for graphical models such as markov chains, dropping the arrows on the edges preserves the independencies, yielding an equivalent undirected graphical model. Similarly, directed trees are fundamentally no different from undirected trees.

An important point to note is that it is the absence of edges that characterizes a graphical model. For any graphical model, it is possible that the compatibility functions (or local conditional probabilities) assume a very special form so that there are more (conditional) independences that hold than what is indicated by the graphical model (which means that some of the edges in the graph could be redundant).

1.2 Inference

In this section, we discuss the problem of determining the marginal distribution $p(\mathbf{x}_{\mathcal{A}})$, the conditional distribution $p(\mathbf{x}_{\mathcal{A}} \mid \mathbf{x}_{\mathcal{B}})$ and the partition function Z, given a graphical model $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ and for any $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$. We will assume that the conditional probability tables (for directed models) or the potential function table (for undirected models) are already known. The sum and product rules of probability yield the following formulae[3] for the marginal, the conditional[4] and the partition function[5] respectively:

$$p(\mathbf{x}_{\mathcal{A}}) = \sum_{\mathbf{x}_{\mathcal{V} \setminus \mathcal{A}}} p(\mathbf{x})$$

$$p(\mathbf{x}_{\mathcal{A}} \mid \mathbf{x}_{\mathcal{O}}) = \frac{p(\mathbf{x}_{\mathcal{A}}, \mathbf{x}_{\mathcal{O}})}{p(\mathbf{x}_{\mathcal{O}})}$$

$$Z = \sum_{\mathbf{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \mathcal{X}_n} \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})$$

All these problems are somewhat similar, in that they involve summation over a very high dimensional space. Computing any of these quantities will involve number of computations that are atleast exponential in the size of $\mathcal{V} \setminus \mathcal{A}$. This is not feasible in many practical applications, where the number of nodes will run into the hundreds or the thousands. As we will see, there is ample redundancy in these computations. We will briefly discuss a simple and pedagogical algorithm called the *elimination algorithm* that provides the intuition as to how the structure of the graph could be exploited to answer some of the questions listed above. More clever algorithms such as the *sum-product algorithm* that captures redundancies more efficiently will be discussed subsequently.

A problem fundamentally different from the three listed above is that of *maximum aposteriori optimization* - determining the mode of a conditional distribution.

$$\widehat{\mathbf{x}}_{\mathcal{A}} = \underset{\mathbf{x}_{\mathcal{A}}}{\operatorname{argmax}}\ p(\mathbf{x}_{\mathcal{A}} \mid \mathbf{x}_{\mathcal{O}})$$

For discrete problems, this is an integer linear programming problem. For general graphs, you cannot do much better than a brute force, whereas, for special graphs (such as trees), this mode can be computed efficiently.

[3]For continuous valued random variables, you can expect the summation to be replaced by integration in each formula.

[4]The problem of computing conditionals is not fundamentally different from the problem of computing the marginals, since every conditional is simply the ratio of two marginals.

[5]The partition function needs to be computed for parameter estimation.

1.2.1 Elimination Algorithm

The elimination algorithm provides a systematic framework for optimizing computations by rearranging sums and products, an instance of which we saw in the proof of theorem 3. Consider the simple directed graph in Figure 1.1. Let us say each random variable takes values from the set $\{v_1, v_2, \ldots, v_k\}$. Brute force computation of $p(x_1 \mid x_5) = \frac{p(x_1,x_5)}{p(x_5)}$ will involve $k \times k^3 = k^4$ computations for the numerator and $k \times k^4 = k^5$ computations for the denominator. To take into account conditioning, we introduce a new potential function $\delta(x_i, x_i')$ which is defined as follows:

$$\delta(x_i, x_i') = \begin{cases} 1 & \text{if } x_i = x_i' \\ 0 & \text{otherwise} \end{cases}$$

and simply write the conditional as

$$p(x_1 \mid x_5) = \frac{p(x_1, x_5)}{p(x_5)} = \sum_{x_2,x_3,x_4,x_5'} p(x_5'|x_3)\delta(x_5, x_5')p(x_3|x_1, x_2)p(x_4|x_2)p(x_2)$$

That is, whenever a variable is observed, you imagine that you have imposed the indicator function for that observation into the joint distribution. Given this simplicification, we will focus on efficiently computing $p(x_1)$, assuming that the δ function will be slapped onto the corresponding factor while computing conditionals.

Using the structure of the graphical model (implicit in the topological ordering over the indices) , we can rearrange the sums and products for $p(x_1|x_5)$

$$p(x_1 \mid x_5) = \left(\sum_{x_2} p(x_2)\left(\sum_{x_3} p(x_3|x_1, x_2)\left(\sum_{x_4} p(x_4|x_2)\right)\left(\sum_{x_5'} p(x_5'|x_3)\delta(x_5, x_5')\right)\right)\right) \tag{1.14}$$

where brackets have been placed at appropriate places to denote domains of summation.

Analysing this computational structure inside-out,

1. We find two innermost factors to be common across all the summations, *viz.*,

$$m_{x_4}(x_2) = \sum_{x_4} p(x_4|x_2)$$

$$m_{x_5}(x_3) = \sum_{x_5'} p(x_5'|x_3)\delta(x_5, x_5')$$

   where m_{x_4} is a *message* function of x_2 and is obtained using $k \times k = k^2$ computations and m_{x_5} is a *message* function of x_3 and similarly obtained using k^2 computations.

2. Further, we can decipher from (1.14), the following message function m_{x_3} of x_1 and x_2 which can be computed using k^3 operations:

$$m_{x_3}(x_1, x_2) = \sum_{x_3} p(x_3|x_1, x_2) m_{x_4}(x_2) m_{x_5'}(x_3)$$

3. Putting all the messages together, we get the message function m_{x_2} of x_1, computable with k^2 operations.

$$m_{x_2}(x_1) = \sum_{x_2} m_{x_3}(x_1, x_2)$$

Thus, the summation over the factorization can be expressed as a flow of message from one node to another; when the message from a node passes on to another, the former gets stripped or *eliminated*. In the step (1), nodes x_4 and x_5 got stripped. In step (2), node x_3 was stripped and finally, in step (3), x_2 got stripped. This yields an elimination ordering[6] $[x_4, x_5, x_3, x_2]$. The order of computation is thus brought down from $O(k^5)$ to $O(max(k^2, k^3)) = O(k^3)$ computations. While some researchers could argue that this may not be a substantial decrease, for larger graphs the gain in speed using this procedure is always substantial.

More formally, consider a root to leaf ordering $\mathcal{I}$ of the nodes, where r is the root node (equivalently, an ordering that corresponds to leaf stripping). Figure 1.1 shows a numbering of the nodes corresponding to such an ordering. We will define as the current active set $\mathcal{A}(k)$, a set of indices of general potential functions. At each step of the algorithm the potential functions are of three different types: (i) some of the local conditional probabilities $p(x_i|\mathbf{x}_{\pi_i})$, (ii) some of the indicators $\delta(x_j, x_j')$ of the observed nodes and (iii) some messages (*c.f.* page 16) generated so far. More formally, the active set of potentials is given by $\{\Psi_\alpha(\mathbf{x}_\alpha)\}_{\alpha \in \mathcal{A}^{(k)}}$, with α being a generic index that ranges over sets of nodes. $\mathcal{A}^{(0)}$ is initialized as the set of all cliques that are associated with potential functions in the graphical model. For example, in Figure 1.1, $\mathcal{A}^{(0)} = \{\{1\}, \{2\}, \{3, 2, 1\}, \{2, 4\}, \{3, 5\}\}$. Then, $\mathcal{A}^{(k)}$ can be computed using the algorithm presented in Figure 1.6.

When the active set construction is complete, the desired conditional/marginal probability can be obtained as

$$p(x_r \mid \mathbf{x}_o) = \frac{\Psi_{\{r\}} x_r}{\sum_{x_r} \Psi_{\{r\}} x_r}$$

A flip side of the elimination algorithm is that it requires a 'good' elimination order to be first determined. The number of elimination orderings is obviously a large value of $(n-1)!$, where n is the number of nodes. Finding a good elimination ordering is an NP hard problem and heuristics have been the only recourse. We will not discuss the elimination algorithm any further, but instead jump to the more efficient sum-product algorithm.

[6]Since either x_4 and x_5 may get stripped first, $[x_5, x_4, x_3, x_2]$ is also a valid elimination ordering.

```
1. Construct an elimination ordering of the nodes so that the target node
at which condition/marginal is desired, is last in the ordering.
2. Initialize $\mathcal{A}^{(0)}$ to the set of all cliques on which potentials are defined.
Set $k = 0$.
for Each $i \in \mathcal{I}$ do
    4. Compute $\Psi_{\beta_i}(\mathbf{x}_{\beta_i}) = \prod_{\{\alpha \in \mathcal{A}^{(k)} | i \in \alpha\}} \Psi_\alpha(\mathbf{x}_\alpha)$ where, $\beta_i = \{i\} \cup \{j \mid \exists \alpha \in \mathcal{A}^{(k)}, \{i,j\} \subset \alpha\}$. That is, $\Psi_{\beta_i}(\mathbf{x}_{\beta_i})$ is a product of potentials
    that have shared factors with $\{i\}$.
    5. Message Computation: Compute the message communicated to
    $x_i$ by stripping out $x_i$ through summing over $x_i$. $M_i(\mathbf{x}_{\gamma_i}) = \Psi_{\gamma_i}(\mathbf{x}_{\gamma_i}) = \sum_{x_i} \Psi_{\beta_i}(\mathbf{x}_{\beta_i})$, where, $\gamma_i = \beta_i \setminus \{i\}$, that is, $\gamma_i$ is the resudial left after strip-
    ping out $\{i\}$ from $\beta_i$ and the message $M_i$ depends only on this residual.
    The computational complexity is determined by the size of the residuals.
    6. Stripping out factors: Remove all $\alpha$ such that $i \in \alpha$ and add $\gamma_i$ to
    the current active set to obtain $\mathcal{A}^{(k+1)}$.
        $\mathcal{A}^{(k+1)} = \mathcal{A}^{(k)} \setminus \{\alpha \in \mathcal{A}^{(k)} \mid i \in \alpha\} \cup \{\gamma_i\}$
end for
```

Figure 1.6: Procedure for constructing the active set, and the message at the desired target node t.

1.2.2 Sum-product Algorithm

The sum-product algorithm builds on the idea of 'messages' as motivated by the elimination algorithm. It involves local computations at nodes, to generate 'messages' which are related by nodes along their edges to their neighbors. This formalism enables simultaneous computation of marginals, conditionals as well as modes for several variable sets. This algorithm generalizes special algorithms such as viterbi, the forward-backward algorithm, kalman filtering, gaussian elimination as well as the fast fourier transform.

We will initially restrict our attention to undirected trees and will later generalize the algorithm. Figure 1.7 shows an example tree structured graphical model. Since the cliques consist of edges and individual nodes, the potential functions are basically either node potentials $\phi_p(x_p)$ or edge potentials $\phi_{p,q}(x_p, x_q)$. The joint distribution for the tree is

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{p \in \mathcal{V}} \phi_p(x_p) \prod_{(p,q) \in \mathcal{E}} \phi_{p,q}(x_p, x_q) \tag{1.15}$$

Note that, all discussions that follow, hold equally well for directed trees, with the special parametrization $\phi_{p,q}(x_p, x_q) = p(x_p|x_q)$ if $x_p \in \pi_{x_q}$ and $\phi_p(x_p) =$

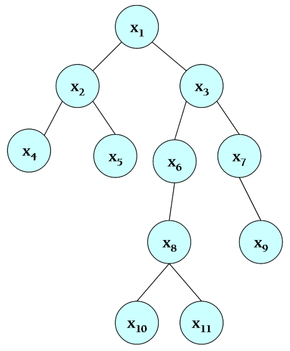


Figure 1.7: A tree graphical model.

$p(x_p)$.

We will deal with conditioning in a manner similar to the way we dealt with conditioning in the case of directed graphical models. For every observed variable $X_o = x_o$, we impose the indicator function $\delta(x_o, x_o')$ onto $\phi_o(x_o')$. Then, for a set of observed variables $\mathbf{X}_{\mathcal{O}}$, the conditional then takes the form:

$$p(\mathbf{x} \mid \mathbf{x}_{\mathcal{O}}) = \frac{1}{Z_{\mathcal{O}}} \prod_{p \in \mathcal{V} \setminus \mathcal{O}} \phi_p(x_p) \prod_{o \in \mathcal{O}} \phi_o(x_o') \delta(x_o, x_o') \prod_{(p,q) \in \mathcal{E}} \phi_{p,q}(x_p, x_q) \tag{1.16}$$

Thus, modifying the compatibility functions appropriately reduces the conditional problem to an instance of the base problem.

The crux of the sum-product algorithm is the following observation on the application of the leaf stripping and message construction procedure in Figure 1.6 to a tree-structured graphical model.

Theorem 4 *When a node i is eliminated, $\gamma_i = \{p\}$, where p is the unique parent of node i. This means, we can write the message as $M_{i \to p}$.*

Proof Sketch: This can be proved by induction on the number of steps in the leaf-stripping (elimination) order. You will need to show that at every step, $\beta_i = \{i, p\}$, where p is the unique parent of i. Consequently, $\gamma_i = \{p\}$ and we can derive the (recursive) expression for $M_{i \to p}(x_p)$ as

$$M_{i \to p}(x_p) = \Psi_{x_p}(x_p) = \underbrace{\sum_{x_i} \phi_i(x_i) \phi_{i,p}(x_i, x_p)}_{SUM} \underbrace{\prod_{q \in \mathcal{N}(i) \setminus p} M_{q \to i}(x_i)}_{PRODUCT} \tag{1.17}$$

Note that the proof of this statement will again involve induction. $\Box$

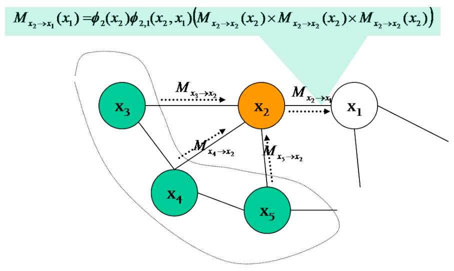

$M_{x_2 \to x_1}(x_1) = \phi_2(x_2)\phi_{2,1}(x_2, x_1)\left(M_{x_2 \to x_2}(x_2) \times M_{x_2 \to x_2}(x_2) \times M_{x_2 \to x_2}(x_2)\right)$
$M_{x_3 \to x_2}$
$M_{x_2 \to x_1}$
$M_{x_4 \to x_2}$
$M_{x_5 \to x_2}$
x3
x2
x1
x4
x5

Figure 1.8: An illustration of (1.17) on a part of a tree structured (undirected) graphical model.

Figure 1.8 illustrates an application of (1.17) on a part of a tree structured graphical model.

Theorem 4 provides a very useful observation; the message is not a function of the vector of all nodes eliminated so far (which could have assumed k^m possible values for m eliminated nodes having k possible values each), but is instead is a function only of the child from which the message is being passed. This allows us to move from the elimination algorithm to the sum-product algorithm. In particular, the parts underlined as 'SUM' and 'PRODUCT' in (1.17) form the basis of the sum-product algorithm. In the sum-product algorithm, at every time step, the computation in (1.17) is performed at every node; each node does local computations and passes 'updated' messages to each of its neighbors. In this way, the computations are not tied to any particular elimination ordering.

Theorem 5 *Consider an undirected tree structured graphical model $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with factorization given by (1.15). Let each random variable X_i, $i \in \mathcal{V}$ assume k possible values from $\mathcal{X} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$. For each edge (u, v), define non-negative messages along both directions: $M_{v \to u}(\alpha_i)$ along $v \to u$ and $M_{u \to v}(\alpha_i)$ along $u \to v$ for each $\alpha_i \in \mathcal{X}$. If r is the iteration number, then the update rule*

$$M_{u\to v}^{(r+1)}(x_v) = \frac{1}{Z_{u\to v}^{(r)}} \underbrace{\sum_{x_u} \phi_u(x_u)\phi_{u,v}(x_u, x_v) \prod_{q \in \mathcal{N}(u)\setminus v} M_{q \to u}^{(r)}(x_u)}_{\widetilde{M_{u\to v}^{(r+1)}}(x_v)}$$

can be executed parallely at every node u along every edge u, v originating at u and will converge, resulting in a fix-point for each $M_{u\to v}^{(r^*)}(x_v)$ for some r^*. That is, there exists an r^* such that $M_{u\to v}^{(r^*+1)}(x_v) = M_{u\to v}^{(r^*)}(x_v)$ for all $(u, v) \in \mathcal{E}$. At convergence,

$$p(x_v) = \phi(x_v) \prod_{u \in \mathcal{N}(v)} M_{u\to v}(x_v) \qquad (1.18)$$

Note that $Z^{(r)}_{u\to v}$ is the normalization factor (required for ensuring numerical stability across iterations but not otherwise) and can be computed at each iteration as

$$Z^{(r)}_{u\to v} = \sum_{x_v} \widetilde{M^{(r+1)}_{u\to v}}(x_v)$$

Theorem 5 does away with the need for any elimination ordering on the nodes and lets computations at different nodes happen in parallel. It leads to the sum-product algorithm[7] for trees, which is summarized in Figure 1.9. The so-called flooding[8] schedule does a 'for' loop at each iteration of the algorithm, for each node in $\mathcal{V}$. By Theorem 5, the procedure will converge. In fact, it can be proved that the algorithm will converge after at most κ iterations, where κ is the diameter (length of the longest path) of $\mathcal{G}$. The intuition is that message passing needs the message from every node to reach every other node and this will take κ iterations for that to happen in the sum-product algorithm.

```
Initialize $M^{(0)}_{u\to v}(x_v)$ for each $(u,v) \in \mathcal{E}$ to some strictly positive random values.
Set $r = 0$.
**repeat**
  **for** Each $u \in \mathcal{V}$ **do**
    **for** Each $v \in \mathcal{N}(u)$ **do**
      $M^{(r+1)}_{u\to v}(x_v) = \frac{1}{Z^{(r)}_{u\to v}} \sum_{x_u} \phi_u(x_u)\phi_{u,v}(x_u,x_v) \prod_{q\in\mathcal{N}(u)\setminus v} M^{(r)}_{q\to u}(x_u)$.
    **end for**
  **end for**
  Set $r = r + 1$.
**until** $M^{(r)}_{u\to v}(x_v) = M^{(r-1)}_{u\to v}(x_v)$ for each $(u,v) \in \mathcal{E}$ and each $x_v \in \mathcal{X}$
Set $r^* = r - 1$.
```

Figure 1.9: The sum-product algorithm for a tree, using the flooding schedule. It converges for $r^* \leq \kappa$, κ being the tree diameter.

There have been modern, interesting uses of message passing techniques on graphs with cycles, such as in the field of sensor networks, where locally parallelizable algorithms such as message passing are highly desirable. While there is nothing that stops us in principle from applying the algorithm in Figure 1.9 to general graphs having cycles, the theory does not guarantee anything at all - neither in terms of convergence nor in terms of the number of iterations. However, in solving partial differential equations, the message passing algorithm is often used. The algorithm is widely used on certain interesting cyclic graphs in the

[7]SIMPLE EXERCISE : Implement the algorithm using Matlab.

[8]The flooding schedule somewhat mimics the flow of water or some liquid through a network; water flows in to a node along an edge and then spreads through the other edges incident on the node.

field of communications. For special problems such as solving linear systems, this method converges[9] on graphs with cycles as well.

However, a schedule such as in Figure 1.9 will typically incur a heavy communication cost, owing to message transmissions. While the flooding schedule is conceptually easy in terms of parallelization, an alternative schedule called the serial schedule is often preferred when parallelization is not of paramount importance. The serial schedule minimizes the number of messages passed. In this schedule, a node u transmits message to node v only when it has received messages from all other nodes $q \in \mathcal{N}(u) \setminus v$. This algorithm will pass a message only once along every direction of each edge (although during different steps). Thus, the scheduling begins with each leaf passing a message to its immediate neighbor. An overhead involved in the serial schedule is that every node needs to keep track of the edges along which it has received messages thus far. For chip level design, the highly parallelizable flooding schedule is always preferred over the serial schedule.

A point to note is that while the algorithm in Figure 1.9 is guaranteed to converge within κ steps, in practice, you might want to run the algorithm for fewer steps, until the messages reasonably converge. This strategy is especially adopted in the belief propagation algorithm, which consists of the following steps at each node of a general graphical model, until some convergence criterion is met:

1. form product of incoming messages and local evidence
2. marginalize to give outgoing message
3. propagate one message in each direction across every link

The belief propagation algorithm will be discussed later.

1.2.3 Max Product Algorithm

The max product algorithm solves the problem of determining the mode or peak of a conditional distribution, specified by a graphical model $\mathcal{G}$, first addressed on page 15.

$$\widehat{\mathbf{x}}_{\mathcal{A}} = \underset{\mathbf{x}_{\mathcal{A}}}{\operatorname{argmax}}\ p(\mathbf{x}_{\mathcal{A}} \mid \mathbf{x}_{\mathcal{O}})$$

where $\mathbf{X}_{\mathcal{O}}$ is the set of observed variables, having observed values $\mathbf{x}_{\mathcal{O}}$, and $\mathbf{X}_{\mathcal{A}}$ are the query variables.

Before looking at the procedure for finding the model of a distribution, we will take a peek at an algorithm for determining the maximum value of $p(\mathbf{x})$, assuming it to be a distribution over an undirected tree $\mathcal{G}$. This algorithm is closely related to the sum product algorithm and can easily be obtained from the sum-product algorithm by replacing the 'SUM' with a 'MAX' in (1.17). This is because, maximums can be pushed inside products and computations can be

[9]For solving linear systems, the message passing algorithm is more efficient than Jacobi, though less efficient than conjugate gradient.

structured in a similar manner as in the sum product algorithm. This places max-product in the league of message passing algorithms.

The sum-product and max-product algorithms are formally very similar. Both methods are based on the distributive law[10]:

- For sum-product: $ab + ac = a(b + c)$.
- For max-product (whenever $a \geq 0$): $\max\,\{ab, ac\} = a \times \max\,\{b, c\}$.

```
Initialize $M^{(0)}_{u\to v}(x_v)$ for each $(u,v) \in \mathcal{E}$ to some strictly positive random values.
Set $r = 0$.
repeat
  for Each $u \in \mathcal{V}$ do
    for Each $v \in \mathcal{N}(u)$ do
      $M^{(r+1)}_{u\to v}(x_v) = \frac{1}{Z^{(r)}_{u\to v}} \max_{x_u} \phi_u(x_u)\phi_{u,v}(x_u, x_v) \prod_{q\in\mathcal{N}(u)\setminus v} M^{(r)}_{q\to u}(x_u)$.
    end for
  end for
  Set $r = r + 1$.
until $M^{(r)}_{u\to v}(x_v) = M^{(r-1)}_{u\to v}(x_v)$ for each $(u,v) \in \mathcal{E}$ and each $x_v \in \mathcal{X}$
Set $r^* = r - 1$.
```

Figure 1.10: The max-product algorithm for a tree, using the flooding schedule. It converges for $r^* \leq \kappa$, κ being the tree diameter.

The max-product and sum-product algorithms can be implemented in a common framework, with a simple flag to toggle between the two. The convergence of the max-product algorithm in Figure 1.10 is very similar to the proof of the convergence of the sum-product algorithm in Figure 1.9 (the proof is rooted in Theorem 5). After convergence of the max-product algorithm in Figure 1.10, the maxium value of $\Pr(\mathbf{X} = \mathbf{x})$ can be retrieved as

$$\max_{\mathbf{x}}\ \Pr(\mathbf{X} = \mathbf{x}) = \max_{x_r}\ \phi(x_r) \prod_{u\in\mathcal{N}(r)} M_{u\to r}(x_r) \tag{1.19}$$

where, we assume that x_r is the root of the tree. However, we need to go beyond the maximum value of a distribution; we need to find the point at which the maximum is attained. To traceback this, and in general to compute $\underset{\mathbf{x}_{\mathcal{A}}}{\text{argmax}}\ p(\mathbf{x}_{\mathcal{A}} \mid \mathbf{x}_{\mathcal{O}})$ we will introduce some additional machinery.

As before (*c.f.* page 19), the conditioning can be obtained as in (1.16).

[10]See commutative semirings for the general algebriac framework. Also refer to work on generalized distributive laws.

Definition 8 *We define the singleton marginal of a distribution $p(\mathbf{x})$ as*

$$\mu_s(x_s) = \frac{1}{\alpha} \max_{\mathbf{x}\setminus\{x_s\}} p(\mathbf{x}) \tag{1.20}$$

and the pairwise marginal as

$$\nu_s(x_s, x_t) = \frac{1}{\alpha} \max_{\mathbf{x}\setminus\{x_s,x_t\}} p(\mathbf{x}) \tag{1.21}$$

where

$$\alpha = \max_{\mathbf{x}} p(\mathbf{x})$$

The max marginals are analogs of marginilization, but with the summation replaced by the max operator over all variables, except x_s in the former or (x_s, x_t) in the latter. While $\mu_s(x_s)$ gives a vector of max-marginals for the variable X_s, $\nu_s(x_s, x_t)$ corresponds to a matrix of values, for the pair of variables (X_s, X_t). You can easily convince yourself that the maximum value of any max-marginal is 1 and it will be attained for atleast one value x_s for each variable X_s.

How can the marginals be tracedback efficiently? And how can they be useful? The marginals encode preferences in the form of sufficient statistics. For example:

$$\mu_1(x_1) = \frac{1}{\alpha} \max_{x_2,x_3,\ldots,x_n} p(\mathbf{x})$$

In fact, we can look at the local maximal configurations, when they are unique and traceback the (unique) global maximal configuration. This is stated in the following powerful theorem.

Theorem 6 *If* $\underset{x_s}{\operatorname{argmax}}\ \mu_s(x_s) = \{x_s^*\}\ \forall s \in \mathcal{V}$ *(that is, there is a unique value of variable X_s that maximizes $\mu_s(x_s)$ for every $s \in \mathcal{V}$), then* $\mathbf{x}^* = \{x_1^*, x_2^*, \ldots, x_n^*\} = \underset{\mathbf{x}}{\operatorname{argmax}}\ p(\mathbf{x})$ *is the unique MAP configuration.*

Proof Sketch: The theorem can be equivalently stated as follows: if $\mu_i(x_i^*) > \mu_i(x_i)$ for all $x_i \neq x_i^*$, and for all $i \in \mathcal{V}$, then $p(\mathbf{x}^*) \geq p(\mathbf{x})$ for all $\mathbf{x} \neq \mathbf{x}^*$. This can be proved by contradiction as follows. Suppose $\mathbf{x}' \in \underset{\mathbf{x}}{\operatorname{argmax}}\ p(\mathbf{x})$. Then, for any $s \in \mathcal{V}$,

$$\mu_s(x_s') = \max_{\mathbf{x}} p(\mathbf{x}) = \max_{x_s} \max_{\mathbf{x}\setminus x_s} p(\mathbf{x}) > \mu_s(x_s)\ \forall\ x_s$$

But by definition

$$\max_{x_s} \max_{\mathbf{x}\setminus x_s} p(\mathbf{x}) = \max_{x_s} \mu_s(x_s) = \{x_s^*\}$$

Thus, $x_s' = x_s^*$. Since $s \in \mathcal{V}$ was arbitrary, we must have $\mathbf{x}' = \mathbf{x}^*$. $\Box$

The singleton max marginal $\mu_s(x_s)$ can be directly obtained as an outcome of the max-product algorithm:

$$\mu_s(x_s) \propto \phi_s(x_s) \prod_{u \in \mathcal{N}(s)} M_{u \to s}(x_s) \tag{1.22}$$

What if $\{x_s^1, x_s^2\} \subseteq \mu_s(x_s)$? That is, if $\mu_s(x_s)$ violates the assumption in theorem 6? Then we have to start worrying about what is happening on the edges, through the medium of the max marginal $\nu_{s,t}(x_s, x_t)$. All we do is randomly sample from the set of configurations that have maximum probability, without caring which one we really get. We first randomly sample from $\underset{x_r}{\text{argmax}}\ \mu_r(x_r)$ at the root r of the tree and then keep randomly sample for a configuration for a child s, given its parent t (*i.e.* for an edge) $\underset{x_s}{\text{argmax}}\ \nu_{st}(x_s, x_t)$ which respects the pairwise coupling. The following theorem states the general traceback mechanism.

Theorem 7 *Given a set of singleton max-marginals, $\{\mu_s \mid s \in \mathcal{V}\}$ and a set of pairwise marginals: $\{\nu_{st} \mid (s,t) \in \mathcal{E}\}$ on a tree $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, $\mathbf{x}^* = (x_1^*, \ldots, x_n^*)$, constructed using the following procedure is a maximal configuration, that is $\mathbf{x}^* \in \underset{\mathbf{x}}{\text{argmax}}\ p(\mathbf{x})$.*

1. *Let r be the root. Let $x_r^* \in \underset{x_r}{\text{argmax}}\ \mu_r(x_r)$.*

2. *In root to leaf order, choose $x_s^* \in \underset{x_s}{\text{argmax}}\ \nu_{st}(x_s, x_t^*)$*

Proof: We will first prove that $\mathbf{x}^*$ is optimal for the root term. For any arbitrary $\mathbf{x}$, by step 1,

$$\mu_r(x_r) \leq \mu_r(x_r^*) \tag{1.23}$$

If $t \to s$ is an edge, then we have by definition of the singleton max-marginal

$$\underset{x_s}{\text{argmax}}\ \nu_{st}(x_s, x_t) = \mu_t(x_t)$$

Thus,

$$\frac{\nu_{st}(x_s, x_t)}{\mu_t(x_t)} \leq 1$$

Since $x_s^* \in \underset{x_s}{\text{argmax}}\ \nu_{st}(x_s, x_t^*)$ by step 2, we must have $\nu_{st}(x_s^*, x_t^*) = \mu_t(x_t^*)$ and therefore, the following upperbound

$$\frac{\nu_{st}(x_s, x_t)}{\mu_t(x_t)} \leq \frac{\nu_{st}(x_s^*, x_t^*)}{\mu_t(x_t^*)} \tag{1.24}$$

for all edges $t \to s$. With repeated application of step 2, we can stitch together the local optimality conditions (1.23) and (1.24) to get

$$\mu_r(x_r) \prod_{t \to s} \frac{\nu_{st}(x_s, x_t)}{\mu_t(x_t)} \leq \mu_r(x_r^*) \prod_{t \to s} \frac{\nu_{st}(x_s^*, x_t^*)}{\mu_t(x_t^*)} \tag{1.25}$$

Just as the singleton max marginals (1.22) can be expressed in terms of the messages from the max product algorithm, the edge max-marginals can also be expressed similarly, by restricting attention to the pair of variables (x_s, X_t) instead of the world of singletons X_s, and by avoiding accounting for the message $M_{s \to t}$ or $M_{t \to s}$ between the pair:

$$\nu_{st}(x_s, x_t) \propto \phi_s(x_s)\phi_t(x_t)\phi_{st}(x_s, x_t) \prod_{u \in \mathcal{N}(s) \setminus t} M_{u \to s}(x_s) \prod_{u \in \mathcal{N}(t) \setminus s} M_{u \to t}(x_t) \tag{1.26}$$

Combining (1.22) with (1.26), we get

$$\frac{\nu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)} \propto \frac{\phi_{st}(x_s, x_t)}{M_{t \to s}(x_s) M_{s \to t}(x_t)} \tag{1.27}$$

Applying (1.27) and (1.25) in the factorization for $p(\mathbf{x})$, we get[11], we obtain

$$p(\mathbf{x}) \leq p(\mathbf{x}^*)$$

Since $\mathbf{x}$ was arbitrary, we must have $\mathbf{x}^* \in \underset{\mathbf{x}}{\text{argmax}}\ p(\mathbf{x})$. $\Box$

An outcome of the proof above is that for trees, the factorization can be written in terms of max-marginals instead of the potential functions:

$$p(\mathbf{x}) \propto \mu_r(x_r) \prod_{t \to s} \frac{\nu_t s(x_s, x_t)}{\mu_t(x_t)}$$

The above form is a directed form of factorization and does not hold for general graphs that may contain cycles. For general graphs, it can be further proved that the following factorization holds

$$p(\mathbf{x}) \propto \prod_{s \in \mathcal{E}} \mu_s(x_s) \prod_{(s,t) \in \mathcal{E}} \frac{\nu_{ts}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)}$$

[11]EXERCISE: Prove.

1.2.4 Junction Tree Algorithm

In many applications, the graphs are not trees. We have not discussed any pricipled techniques for finding marginals and modes for graphs that are not trees, though we have discussed them for trees. The elimination algorithm discussed earlier is applicable for some general graphs, but the question of what elimination should be chosen, needs to be addressed. The junction tree algorithm is very much related to the elimination algorithm, but is a more principled approach for inferencing in directed acyclic graphs. Like the sum-product algorithm, the junction tree algorithm can 'recycle' computations. This is unlike the general elimination algorithm, whose computation was focused completely on a single node. The work on junction trees is primarily attributed to Lauritzen and Spielgelhalter (1998). The correspondence between the graph-theoretic aspect of locality and the algorithmic aspect of computational complexity is made explicit in the junction tree framework.

For a graph with nodes, it is an intuitive idea to consider clustering completely connected nodes, form a tree connecting these clusters and finally perform message passing the tree. This idea can be formalized using the concept of a *clique tree*.

Definition 9 *Given a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with a set $\Pi \subseteq 2^{\mathcal{V}}$ of maximal cliques, a clique tree $\mathcal{T}_{\mathcal{G}}$ is a tree, whose vertices correspond the maximal cliques Π of $\mathcal{G}$ and such that there is an edge between two nodes in the tree only if*[12] *there is an edge in $\mathcal{G}$ between two nodes across the corresponding maximal cliques.*

Let us take some examples:

- For the acylcic graph $\mathcal{G}$ with $\mathcal{V} = \{1,2,3,4,5,6\}$, $\mathcal{E} = \{(1,2),(1,3),(2,4),(2,5),(3,6)\}$, the (not so interesting) clique tree $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{E}_{\Pi} >$ would be $\Pi = \{[1,2],[1,3],[2,4],[2,5],[3,6]\}$ and $\mathcal{E}_{\Pi} = \{([1,2],[2,4]),([1,2],[2,5]),([1,2],[1,3]),([1,3],[3,6])\}$.

- For the cyclic graph $\mathcal{G}$ with $\mathcal{V} = \{1,2,3,4\}$, $\mathcal{E} = \{(1,2),(1,3),(2,4),(3,4),(2,3)\}$, the clique tree $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{E}_{\Pi} >$ would have $\Pi = \{[1,2,3],[2,3,4]\}$ and $\mathcal{E}_{\Pi} = \{([1,2,3],[2,3,4])\}$. We will adopt the practice of labeling the edge connecting nodes corresponding to two maximal cliques $\mathcal{C}_1$ and $\mathcal{C}_2$, with their intersection $\mathcal{C}_1 \cap \mathcal{C}_2$, which will be called the *separator set*. In the second example here, the separator set for vertices $[1,2,3]$ and $[2,3,4]$ is $[2,3]$.

- For the slightly different cyclic graph $\mathcal{G}$ with $\mathcal{V} = \{1,2,3,4\}$, $\mathcal{E} = \{(1,2),(1,3),(2,4),(3,4)\}$, there are many possible clique trees. One possible clique tree $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{E}_{\Pi} >$ would have $\Pi = \{[1,2],[1,3],[2,4],[3,4]\}$ and $\mathcal{E}_{\Pi} = \{([1,2],[1,3]),([1,2],[2,4]),([1,3],[3,4])\}$. It is a bit worrisome here that $[2,4]$ and $[3,4]$ are not connected, since a myopic or 'local' sum-product algorithm, running on the clique tree might make a wrong inference that $[2,4]$ and $[3,4]$ do not share anything in common. In this example, for instance, the message coming from $[2,4]$,

[12]Since we are interested in a clique 'tree', it may be required to drop certain edges in the derived graph. This can be seen through the third example.

through $[1, 2]$ to $[3, 4]$ has marginalized over X_4. But this is incorrect, since $[3, 4]$ include the variable X_4.

With the last example in mind, we will refine the notion of a clique tree to a *junction tree*, in order to ensure that local computations are guaranteed to produce globally consistent answers; **that different copies of the same random variable have ways of communicating with each other**.

Definition 10 *A junction tree for a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ having a set $\Pi \subseteq 2^{\mathcal{V}}$ of maximal cliques, is a particular type of clique tree $\mathcal{T}_{\mathcal{G}}$ such that for any two $\mathcal{C}_1, \mathcal{C}_2 \in \Pi$, $\mathcal{C}_1 \cap \mathcal{C}_2$ is a subset of every separator set on the unique path from $\mathcal{C}_1$ to $\mathcal{C}_2$ in $\mathcal{T}_{\mathcal{G}}$. This property of junction trees is called the running intersection property.*

Based on this definition, the clique trees for the first two examples are junction trees, whereas that for the third is not a junction tree. In fact, there are no junction tree for the third example. Let us consider another example.

- Consider the cyclic graph $\mathcal{G}$ with $\mathcal{V} = \{1, 2, 3, 4, 5\}$, $\mathcal{E} = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5)\}$. There are many possible clique trees for $\mathcal{G}$. One possible clique tree $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{E}_{\Pi} >$ has $\Pi = \{[1, 2, 3], [2, 3, 4], [3, 4, 5]\}$ and $\mathcal{E}_{\Pi} = \{([1, 2, 3], [2, 3, 4]), ([2, 3, 4], [3, 4, 5])\}$ and this happens to also be a junction tree. Another clique tree with same vertex set Π and $\mathcal{E}'_{\Pi} = \{([1, 2, 3], [3, 4, 5]), ([2, 3, 4], [3, 4, 5])\}$ is not a junction tree, since node 2 which is in the intersection of $[1, 2, 3]$ and $[2, 3, 4]$ is not on every separator on the path between these two nodes. This illustrates that how you generate your clique tree matters; some clique trees may happen to be junction trees, while some may not.

- For the cyclic graph $\mathcal{G}$ with $\mathcal{V} = \{1, 2, 3, 4\}$, $\mathcal{E} = \{(1, 2), (1, 3), (2, 4), (3, 4)\}$, considered in the third example above, there are no junction trees possible.

The global picture on a graph is specified by the factorization property:

$$p(\mathbf{x}) \propto \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \tag{1.28}$$

On the other hand, the local message passing algorithm should honor constraints set by the separator set; that the configuration of variables in the separator set are the same in both its neighbors. More precisely, if we define the set $\widetilde{\mathbf{x}}_{\mathcal{C}} = \{\widetilde{x}_{i,\mathcal{C}} | i \in \mathcal{C}\}$ for each $\mathcal{C} \in \Pi$, then, for each separator set $\mathcal{C}_1 \cap \mathcal{C}_2$, the separator sets define the constraints in the form

$$\prod_{i \in \mathcal{C}_1 \mathcal{C}_2} \delta\left(\widetilde{x}_{i,\mathcal{C}_1} = \widetilde{x}_{i,\mathcal{C}_2}\right) \tag{1.29}$$

The factorization that message passing on $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{C}' >$ should see with the set of additional constraints will involve multiple copies of each random variable, but will be tied together through the δ constraints in (1.29):

$$\widetilde{p}(\widetilde{x}_{i,\mathcal{C}},\ \forall i \in \mathcal{C},\ \forall \mathcal{C} \in \Pi) \propto \prod_{\mathcal{C}\in\Pi} \phi_{\mathcal{C}}(\widetilde{\mathbf{x}}_{\mathcal{C}}) \prod_{(\mathcal{C}_1,\mathcal{C}_2)\in\mathcal{E}_\Pi} \prod_{i\in\mathcal{C}_1\cap\mathcal{C}_2} \delta\left(\widetilde{x}_{i,\mathcal{C}_1} = \widetilde{x}_{i,\mathcal{C}_2}\right) \quad (1.30)$$

The (localized) sum-product algorithm will work precisely on the junction tree $\mathcal{T}_{\mathcal{G}}$ with factorization as specified in (1.30). The factorization in (1.30) is in fact equivalent to the factorization in (1.28). In (1.30), the multiple copies of x_i's cancel out aginst the constraints δ constraint. The is called the junction tree property and is formally stated in the next proposition.

Theorem 8 *For a graph $\mathcal{G}$ having distribution $p(\mathbf{x})$ as in (1.28) and for its junction tree $\mathcal{T}_{\mathcal{G}}$ having distribution $\widetilde{p}$ specified by (1.30), the $\widetilde{p}$ distribution satisfies the following property:*

$$\widetilde{p}(\widetilde{x}_{i,\mathcal{C}},\ \forall i \in \mathcal{C},\ \forall \mathcal{C} \in \Pi) = \begin{cases} p(\mathbf{x}) & \textit{if } \{x_{i,\mathcal{C}_1} = x_{i,\mathcal{C}_2} \,|\forall \mathcal{C}_1, \mathcal{C}_2 \in \Pi,\ \forall\ i \in \mathcal{C}_1 \cap \mathcal{C}_2\} \\ 0 & \textit{otherwise} \end{cases}$$

That is, the new distribution $\widetilde{p}$ is faithful to the original distribution. The transitivity due to the running intersection property of junction trees is exactly what you need for this desirable property to hold.

The proof of this theorem is trivial, given the junction tree assumption. As an exercise, you may verify the truth of this statement for all the junction tree examples considered so far. The message passing formalisms in Figures 1.9 and 1.10, when applied to the junction tree, will land up not accepting contributions from inconsistent configurations, owing to the δ constraint and will therefore discover the true marginal/mode.

Theorem 9 *Suppose that $\mathcal{G}$ has a junction tree $\mathcal{T}_{\mathcal{G}} =< \Pi, \mathcal{E}_\Pi >$. Then running the sum-product or max-product algorithm on the distribution $\widetilde{p}$ defined in (1.28) will output the correct marginals or modes respectively, for p defined for $\mathcal{G}$, in (1.30). The $\phi_{\mathcal{C}}(\widetilde{\mathbf{x}}_{\mathcal{C}})$ can be thought of as node potentials for $\mathcal{T}_{\mathcal{G}}$, while $\delta\left(\widetilde{x}_{i,\mathcal{C}_1} = \widetilde{x}_{i,\mathcal{C}_2}\right)$ are the edge potentials.*

Theorem 9 presentes a transformation of the original problem to a problem on a right kind of tree on which the running intersection property holds so that marginals and modes are preserved. The proof of this theorem is also simple. In practice, the message passing algorithm need not create multiple copies of the shared variables; the sharing can be imposed implicitly.

1.2.5 Junction Tree Propagation

The *junction tree propagation algorithm* is the sum-product algorithm applied to the junction tree, with factorization specified by (1.30). It is due to Shafer and Shenoy [SS90]. Consider the message update rule from the algorithm in Figure 1.9.

$$M_{u\to v}^{(r+1)}(x_v) = \frac{1}{Z_{u\to v}^{(r)}} \sum_{x_u} \phi_u(x_u)\phi_{u,v}(x_u,x_v) \prod_{q\in\mathcal{N}(u)\setminus v} M_{q\to u}^{(r)}(x_u)$$

If neighbors u and v are replaced by neighboring cliques $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively, the equation becomes

$$M_{\mathcal{C}_1\to\mathcal{C}_2}^{(r+1)}(\mathbf{x}_{\mathcal{C}_2}) = \frac{1}{Z_{\mathcal{C}_1\to\mathcal{C}_2}^{(r)}} \sum_{\mathbf{x}'_{\mathcal{C}_1}} \phi_{\mathcal{C}_1}(\mathbf{x}'_{\mathcal{C}_1}) \underbrace{\prod_{i\in\mathcal{C}_1\cap\mathcal{C}_2} \delta\left(x'_{i,\mathcal{C}_1} = x_{i,\mathcal{C}_2}\right)}_{based\ on\ separator\ set\ \mathcal{C}_1\cap\mathcal{C}_2} \prod_{\mathcal{C}_3\in\mathcal{N}(\mathcal{C}_1)\setminus\mathcal{C}_2} M_{\mathcal{C}_3\to\mathcal{C}_1}^{(r)}(\mathbf{x}'_{\mathcal{C}_1})$$

The constraint based on the separator set ensures that configurations that are not consistent do not contribute to the outermost summation $\sum\limits_{\mathbf{x}'_{\mathcal{C}_1}}$. The expression for the message can therefore be equivalently written as

$$M_{\mathcal{C}_1\to\mathcal{C}_2}^{(r+1)}(\mathbf{x}_{\mathcal{C}_2}) = \frac{1}{Z_{\mathcal{C}_1\to\mathcal{C}_2}^{(r)}} \sum_{\mathbf{x}'_{\mathcal{C}_1\setminus\mathcal{C}_2}} \phi_{\mathcal{C}_1}(\mathbf{x}'_{\mathcal{C}_1\setminus\mathcal{C}_2}, \mathbf{x}_{\mathcal{C}_2}) \prod_{\mathcal{C}_3\in\mathcal{N}(\mathcal{C}_1)\setminus\mathcal{C}_2} M_{\mathcal{C}_3\to\mathcal{C}_1}^{(r)}(\mathbf{x}'_{\mathcal{C}_1}) \tag{1.31}$$

Note that in (1.31), the constraints based on separator sets are implicitly captured in the summation $\sum\limits_{\mathbf{x}'_{\mathcal{C}_1\setminus\mathcal{C}_2}}$ over only a partial set of variables from $\mathbf{x}'_{\mathcal{C}_1}$. Further, the message $M_{\mathcal{C}_1\to\mathcal{C}_2}^{(r+1)}(\mathbf{x}_{\mathcal{C}_2})$ is not a function of the complete vector $\mathbf{x}_{\mathcal{C}_2}$ but is only a function of $\mathbf{x}_{\mathcal{C}_2\cap\mathcal{C}_1}$. Rewriting (1.31) to reflect this finding, we have

$$M_{\mathcal{C}_1\to\mathcal{C}_2}^{(r+1)}(\mathbf{x}_{\mathcal{C}_2\cap\mathcal{C}_1}) = \frac{1}{Z_{\mathcal{C}_1\to\mathcal{C}_2}^{(r)}} \sum_{\mathbf{x}'_{\mathcal{C}_1\setminus\mathcal{C}_2}} \phi_{\mathcal{C}_1}(\mathbf{x}'_{\mathcal{C}_1\setminus\mathcal{C}_2}, \mathbf{x}_{\mathcal{C}_2\cap\mathcal{C}_1}) \prod_{\mathcal{C}_3\in\mathcal{N}(\mathcal{C}_1)\setminus\mathcal{C}_2} M_{\mathcal{C}_3\to\mathcal{C}_1}^{(r)}(\mathbf{x}'_{\mathcal{C}_1\cap\mathcal{C}_3}) \tag{1.32}$$

This finding is important, since it helps reduce the computational complexity of the algorithm. You need to send messages whose sizes do not depend on the size of the cliques themselves but only on the size of the separator sets. Thus, if each variable was multinomial with k possible values, then the message size would be $k^{|\mathcal{C}_1\cap\mathcal{C}_2|}$ instead of $k^{|\mathcal{C}_2|}$. Thus, the complexity of junction tree propagation is exponential in the size of the separator sets. Typically however, the size of seperator sets are not much smaller than the cliques themselves.

The junction tree propagation will converge, by the convergence property of the sum-product algorithm. After convergence, the marginals can be obtained as

$$p(\mathbf{x}_{\mathcal{C}}) = \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \prod_{\mathcal{D}\in\mathcal{N}(\mathcal{C})} M_{\mathcal{D}\to\mathcal{C}}(\mathbf{x}_{\mathcal{C}\cap\mathcal{D}}) \tag{1.33}$$

1.2.6 Constructing Junction Trees

How do we obtain a junction tree for a graph. And what classes of graphs have junction trees? We already saw an example of a graph that did not have any junction tree; $\mathcal{G} =< \mathcal{V},\mathcal{E} >$, $\mathcal{V} = \{1,2,3,4\}$, $\mathcal{E} = \{(1,2),(1,3),(2,4),(3,4)\}$. We also saw an example for which a particular clique tree was not a junction tree, though it had another clique tree that was a junction tree: $\mathcal{G}'$ with $\Pi = \{1,2,3,4,5\}$, $\mathcal{E}_\Pi = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5)\}$. The junction tree $< \mathcal{V}_t', \mathcal{E}_t >$ has $\mathcal{V}_t' = \{[1,2,3],[2,3,4],[3,4,5]\}$ and $\mathcal{E}_t' = \{([1,2,3],[2,3,4]),([2,3,4],[3,4,5])\}$, which corresponds to the elimination ordering $[1,3,2,4,5]$. While the clique tree $\mathcal{V}_t'' = \mathcal{V}_t'$ and $\mathcal{E}_t'' = \{([1,2,3],[3,4,5]),([2,3,4],[3,4,5])\}$ is not a junction tree and corresponds to the elimination ordering $[1,2,4,5,3]$. We can see that the construction of the junction tree really depends on the choice of a 'nice' elimination ordering. One difference between $\mathcal{G}$ and $\mathcal{G}'$ is that, while in the former, you cannot eliminate any node without adding additional edges, in the latter, you have an elimination ordering $[1,3,2,4,5]$ that does not need to add extra edges. For $\mathcal{G}'$, the elimination ordering $[1,2,3,4,5]$ will not yield a junction tree.

This leads to the definition of a triangulated graph, one of the key properties of any graph which can be transformed into a junction tree. In fact, a requirement will be that an elimination algorithm is 'good' for junction tree construction only if it leads to a triangulated graph.

Definition 11 *A cycle is chordless if no two non-adjacent vertices on the cycle are joined by an edge. A graph is triangulated it is has no chordless cycles.*

Thus, the graph $\mathcal{G}$ with $\mathcal{V} = \{1,2,3,4\}$, $\mathcal{E} = \{(1,2),(1,3),(2,4),(3,4)\}$ is not triangulated, whereas, the graph $\mathcal{G}'$ with $\Pi' = \{1,2,3,4,5\}$, $\mathcal{E}_\Pi' = \{(1,2),(1,3),(2,3),(2,4),(3,4),(3,5)\}$ is triagulated. In fact, every triangulated graph has at least one junction tree. Another equivalent characterization of a triangulated graph is as a *decomposable graph.*

Definition 12 *A graph $\mathcal{G} =< \mathcal{V},\mathcal{E} >$ is decomposable either if it is complete or if $\mathcal{V}$ can be recursively divided into three disjoint sets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{S}$ such that*

1. *$\mathcal{S}$ separates $\mathcal{A}$ and $\mathcal{B}$ and*
2. *$\mathcal{S}$ is fully connected (i.e., a clique).*
3. *$\mathcal{A}\cup\mathcal{S}$ and $\mathcal{B}\cup\mathcal{S}$ are also decomposable.*

Following are examples of decomposable graphs:

- $\mathcal{V} = \{1,2,3\}$, $\mathcal{E} = \{(1,2),(2,3)\}$.
- $\mathcal{V} = \{1,2,3,4\}$, $\mathcal{E} = \{(1,2),(2,3),(3,4),(4,1),(2,4)\}$. Application of elimination procedure on this graph, say starting with 3 should lead in 2 and 4 being connected together, which are already connected in this graph. This shows the connection between decomposable graphs and elimination.

However, for $\mathcal{V} = \{1,2,3,4\}$ and $\mathcal{E} = \{(1,2),(2,3),(3,4),(4,1)\}$, $\mathcal{G}$ is not decomposable.

Recall from Section 1.2.1, the elimination algorithm, which eliminated one node at a time, while connecting its immediate neighbors. Thus, for any undirected graph, by the very construction of the elimination algorithm, it is obvious that the reconstituted graph, output by the elimination algorithm is always triangulated. This can be proved by induction on the number of vertices in the graph[13]. The statement is trivial for the base case of a one node graph.

The following theorem is a *fundamental characterization* of graphs that have junction trees.

Theorem 10 *The following are equivalent ways of characterizing a graph $\mathcal{G}$:*

1. *$\mathcal{G}$ is decomposable.*
   - *(This captures the 'divide-and-conquer' nature of message passing algorithms. In fact, the message passing algorithm exploited*

   a divide and conquer strategy for computation on trees.)
2. *$\mathcal{G}$ is triangulated.*
   - *(Elimination can result in a triangulated graph.)*
3. *$\mathcal{G}$ has a junction tree.*
   - *(If the graph is triangulated, it must have at least one junction tree. And junction tree is a good canonical data structure for conducting computations on general graphs.)*

Some practical impacts of this theorem are listed itemized in brackets by the side of each of the equivalent characterizations of $\mathcal{G}$. The equivalence of the first and second statements in the theorem can be proved very simply by induction on the number of nodes in the graph.

The first step in the junction tree algorithm is triangulating a graph. This might mean adding extra edges or increasing clique size. But this cannot be harmful[14], since the potential function can be defined over a larger clique as the

[13]Prove: EXERCISE.

[14]What characterizes a graphical models is not the presence of edges, but the absence of edges. As an the extreme example, a completel graph potentially subsumes every graphical model.

product of potential functions over its sub-parts. Given a triangulated graph, we know that it must have a junction tree by virtue of theorem 10. How can a junction tree be constructed from a triangulated graph? The first step would be to isolate all its cliques. Going back to the second example on page 28, the triangulated graph $\mathcal{G}$ with $\mathcal{V} = \{1, 2, 3, 4, 5\}$, $\mathcal{E} = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5)\}$ has three maximal cliques: $\Pi = \{[1, 2, 3], [2, 3, 4], [3, 4, 5]\}$. There are different ways to connect up the cliques to form a tree. One possible clique tree $\mathcal{T}_\mathcal{G} =< \Pi, \mathcal{E}_\Pi >$ has $\mathcal{E}_\Pi = \{([1, 2, 3], [2, 3, 4]), ([2, 3, 4], [3, 4, 5])\}$ and this happens to also be a junction tree. Another clique tree has $\mathcal{E}'_\Pi = \{([1, 2, 3], [3, 4, 5]), ([2, 3, 4], [3, 4, 5])\}$ is not a junction tree, since node 2 which is in the intersection of $[1, 2, 3]$ and $[2, 3, 4]$ is not on every separator on the path between these two nodes.

While you can discover a junction tree by exhaustive search, that is an infeasible idea.However, the search for a junction tree can be performed efficiently by making use of the following theorem.

Theorem 11 *Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a triangulated graph with $\mathcal{V} = \{X_1, X_2, \ldots, X_n\}$ and with Π being the set of maximal cliques. Let $\mathcal{T}_\mathcal{G}$ be a spanning tree for the clique graph of $\mathcal{G}$, having $\Pi = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$ as the set of vertices and $\mathcal{E}_\Pi = \{e_1, e_2, \ldots, e_{m-1}\}$ as the set of edges ($|\Pi| = m$ and $|\mathcal{E}_\Pi| = m - 1$). Let $\mathcal{S}(e)$ be the separator associated*[15] *with any edge $e \in \mathcal{E}_\Pi$. We will define the weight*[16] *of the spanning tree as*

$$w(\mathcal{T}_\mathcal{G}) \in \sum_{i=1}^{m-1} |\mathcal{S}(e_i)| = \sum_{i=1}^{m-1} \sum_{j=1}^{n} [X_j \in \mathcal{S}(e_i)] \tag{1.34}$$

where [condition] is the indicator function that assumes value 1 if and only if condition is satisfied and is 0 otherwise. Then, if

$$\widehat{\mathcal{T}_\mathcal{G}} = \underset{\mathcal{T}_\mathcal{G}}{\text{argmax}}\ w(\mathcal{T}_\mathcal{G})$$

$\widehat{\mathcal{T}_\mathcal{G}}$ must be a junction tree.

Proof: First we note that for any variable X_j, the number of separator sets in $\mathcal{T}_\mathcal{G}$ in which X_j appears is upper bounded by the number of cliques in which X_j appears.

$$\sum_{i=1}^{m-1} [X_j \in \mathcal{S}(e_i)] \leq \sum_{i=1}^{m} [X_j \in \mathcal{C}_i] \tag{1.35}$$

[15]Since any edge e in the cliqe graph is of the form $(\mathcal{C}_1, \mathcal{C}_2)$, where $\mathcal{C}_1, \mathcal{C}_2 \in \Pi$, the separator associated with e can be viewed as a function $\mathcal{S}(e) = \mathcal{C}_1 \cap \mathcal{C}_2$.

[16]For the example above with $\mathcal{E}_\Pi = \{([1, 2, 3], [2, 3, 4]), ([2, 3, 4], [3, 4, 5])\}$, the weight is $2 + 2 = 4$, and this also happens to be a junction tree. For $\mathcal{E}'_\Pi = \{([1, 2, 3], [3, 4, 5]), ([2, 3, 4], [3, 4, 5])\}$, the weight is $1 + 2 = 3$ and this is not a junction tree.

Equality will hold if and only if the running intersection property holds for X_j in $\mathcal{T}_\mathcal{G}$. By interchanging the order of summations in (1.34) and applying the inequality in (1.35), it follows that

$$w(\mathcal{T}_\mathcal{G}) = \sum_{i=1}^{m-1}\sum_{j=1}^{n}[X_j \in \mathcal{S}(e_i)] \le \sum_{j=1}^{n}\left(\sum_{i=1}^{m}[X_j \in \mathcal{C}_i] - 1\right)$$

Interchanging the summations in the rightmost term yields an upper-bound on $w(\mathcal{T}_\mathcal{G})$, which can be attained if and only if $\mathcal{T}_\mathcal{G}$ is a junction tree (that is, if and only if equality holds in (1.35) for all $1 \le j \le n$)

$$w(\mathcal{T}_\mathcal{G}) \le \sum_{i=1}^{m}|\mathcal{C}_i| - n$$

We know from lemma 10 that if $\mathcal{G}$ is triangulated, it must have a junction tree. Given that $\mathcal{G}$ is triangulated, $\widehat{\mathcal{T}}_\mathcal{G} = \underset{\mathcal{T}_\mathcal{G}}{\text{argmax}}\ w(\mathcal{T}_\mathcal{G})$ must be a junction tree and will satisfy $w(\widehat{\mathcal{T}}_\mathcal{G}) = \sum_{i=1}^{m}|\mathcal{C}_i| - n$. $\Box$

The maximum weight spanning tree problem in (1.34) can be solved exactly by executing the following step $m-1$ times, after intializing $\mathcal{E}_\Pi^{all}$ to all possible 'legal' edges between nodes in Π and $\mathcal{E}_\Pi = \{\}$

1. For $i = 2$ to $m-1$, if

   $$\widehat{e} = \underset{e \in acyclic(\mathcal{E}_\Pi, \mathcal{E}_\Pi^{all})}{\text{argmax}} |\mathcal{S}(e)|$$

   then set $\mathcal{E}_\Pi = \mathcal{E}_\Pi \cup \{e\}$ and $\mathcal{E}_\Pi^{all} = \mathcal{E}_\Pi^{all} \setminus \{e\}$.

Here, $acyclic(\mathcal{E}_\Pi, \mathcal{E}_\Pi^{all}) = \left\{e \in \mathcal{E}_\Pi^{all}\, |\mathcal{E}_\Pi \cup \{e\}\ \textit{has no cycles}\right\}$. This can be efficiently implemented using Kruksal and Prim's algorithm. The only additional requirement is that this problem requires specialized data structure to quickly check if $e \in acyclic(\mathcal{E}_\Pi, \mathcal{E}_\Pi^{all})$, that is, if addition of e to the current set of edges would induce any cycle. This discussion is also relevant for learning tree structured graphical models such that the structure maximizes some objective function on the data.

In Figure 1.11, we present the overall junction tree algorithm. Typically, junction tree propagation is the most expensive step (and is the main 'online' step) and has complexity $O\left(m|\mathcal{C}_{max}|^k\right)$, where k is the maximum number of states for any random variable and $\mathcal{C}_{max}$ is the largest clique. The treewidth τ of a graph is defined as $\tau = \mathcal{C}_{\max} - 1$ in the optimal triangulation. Thus, the junction tree propagation algorithm scales exponentially in the treewidth τ. There are many elimination orderings/triangulations. The best triangulation is the one that leads to smallest value of $\mathcal{C}_{max}$. The problem of finding the best elimination ordering or of finding the best junction tree is NP-hard. In

Input: A graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$.
Pre-processing:

1. **Moralization**: Convert a directed graph to an undirected graph by connecting parents.
2. Introduce delta functions for observed variables.

Triangulation: Triangulate the graph.
Junction tree construction: Construct a junction tree from the triangulated graph.
Junction tree propagation: Using sum-product or max-product, propagate messages on the junction tree.

Figure 1.11: The high-level view of the Junction tree algorithm.

practice, there are many heuristics that work well. Though NP-hard in a worst case sense, this problem is much easier in the average case.

Many problems do not have bounded treewidth. The junction tree algorithm is limited in its application to families of graphical models that have bounded treewidth. Many common graphical models, such as grid structured graphical model that is commonplace in image processing have very high treewidth. The treewidth of an $n \times n$ grid (*i.e.*, n^2 nodes) scales as $O(n)$. Thus, junction tree becomes infeasible for grids as large as 500×500, though it is applicable in theory.

1.2.7 Approximate Inference using Sampling

While the generic junction tree method is principled, it is limited to graphs with bounded treewidth. There are several approximate inference methods that could be considered as alternatives, in practice. One class of approximate inference techniques is the class of sampling methods.

Monte Carlo Methods

The general umbrella problem underlying Monte Carlo sampling methods is

$$E[f] = \int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \tag{1.36}$$

where $f(\mathbf{x})$ is some function defined on some possibly high dimensional space in $\Re^n$ and p is a distribution defined on the same space. For $f(\mathbf{x}) = \mathbf{x}$, $E[f]$ turns out to be the mean. For $f(\mathbf{x}) = \delta(\mathbf{x} = \mathbf{x}')$, $E[f]$ becomes the marginal probability $p(\mathbf{X} = \mathbf{x}')$ or more general, for $f(\mathbf{x}) = \delta(\mathbf{x} \leq \mathbf{x}')$, $E[f]$ becomes the tail probability $p(\mathbf{x} \geq \mathbf{x}')$. The goal of sampling methods, such as Monte

Carlo methods is to approximate such integrals over such possibly high dimensional space using sampling techniques. If we could collect iid samples[17] $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)})$ from $p(.)$, then

$$\widehat{f} = \frac{1}{m}\sum_{i=1}^{m} f(\mathbf{x}^{(i)})$$

is a Monte Carlo estimate of $E[f]$. Since $\overline{\mathbf{x}}$ is a collection of random samples from $p(.)$, $\widehat{f}$ is also a random variable. We could ask some questions about such an estimator:

- Is $\widehat{f}$ unbiased? That is, on an average, will the estimator produce the right quantity? The estimator is unbiased if

  $$E[f] = E_{\overline{\mathbf{x}}}[\widehat{f}]$$

  Using the linearity property of expectation,

  $$E_{\overline{\mathbf{x}}}[\widehat{f}] = \frac{1}{m}\sum_{i=1}^{m} E\left[f(\mathbf{x}^{(i)})\right] = E[f]$$

  that is, the estimator $\widehat{f}$ is indeed unbiased and gives the right answer on an average.

- The unbiased requirement on the part of the estimator only requires the sample mean to match the expected mean, on an average. It may also be desirable that the variance be stable. A related expecation is that as the number m of samples increases, the estimate should get closer to the actual answer. In fact, this is true for the estimator just discussed. It can be shown that if f has finite variance,

  $$var(\widehat{f}) = \frac{1}{m^2}\sum_{i=1}^{m} var\left(f(\mathbf{x}^{(i)})\right) = \sum_{i=1}^{m} \frac{var\left(f(\mathbf{x}^{(i)})\right)}{m} \qquad (1.37)$$

  or equivalently, the spread for $var(\widehat{f})$ is $\frac{1}{\sqrt{m}}$ times the spread for $\widehat{f}$. From this, we can infer that as $m \to \infty$, $var(\widehat{f}) \to \infty$.

The discussion thus far was centered around the assumption that we can draw samples from $p(.)$. This area of sampling has warranted seperate attention for research. Even generation of pseudo random numbers is not straightforward[18]. As another example, it is not straightforward to sample efficiently from

[17]The difference between $\overline{\mathbf{x}}$ here and in the case of maximum likelihood estimation is that in the latter case, $\overline{\mathbf{x}}$ is data provided, whereas here, we consider $\overline{\mathbf{x}}$ sampled from the model itself. However, the analytical form is similar.

[18]If a distribution function can be 'inverted' a common strategy is to pass a unform distribution through it to generate samples.

a typically graphical model-like distribution (especially if it is multi-model) such as

$$p(x) = \frac{1}{Z}\underbrace{\exp\left\{ax^4 + bx^3 + cx^2 + dx + e\right\}}_{\wp(x)}$$

where, the $\wp(x)$ part is easy to compute, whereas $\frac{1}{Z}$ is hard to compute.

We will not look at a series of attempts at sampling from a distribution $p(.)$.

Adopting Numeric Methods

One natural way out is to adopt standard numerical methods, such as the standard numerical recipe for evaluating an integral from first principles - creating discrete intervals and then letting the intervals shrink.

1. Discretize the space of x (such as the real line) into k discrete points, $x_1, x_2, \ldots, x_k$

2. Compute an approximation to z as $\widehat{z} = \sum_{i=1}^{k} \wp(x_i)$.

3. The samples can then be drawn from one of k points based on the distribution p_d:

$$p_d(x_i) = \frac{\wp(x_i)}{\widehat{z}}$$

   The new distribution has point masses at the samples $x_1, x_2, \ldots, x_k$. As the number of grows larger, the approximation will get better

While a controllable approximation that works well in one dimension, how well does such a discretization method scale to higher dimensions. The number of discrete points scales exponentially[19] in the number of dimensions as k^n (n being the dimensionality). Thus, discretization is not feasible in higher dimensions. Another factor in favour of Monte Carlo methods, vis-a-vis numerical techniques is that the statement (1.37) for the Monte Carlo estimator is really independent of the dimensionality n.

Rejection Sampling

Rejection sampling, which dates back to von Neumann in 1950's assumes that in addition to the decomposition $p(\mathbf{x}) = \frac{z}{\wp(\mathbf{x})}$ with $\wp(x)$ easy to compute and $\frac{1}{Z}$ is hard to compute, we also have a proposal distribution $q(\mathbf{x})$ that is relatively easy to (exactly) sample from. q could be one of Gaussians, Cauchy, or some other member of the exponential family.

$$q(\mathbf{x}) = \frac{z_q}{\widehat{q}(\mathbf{x})}$$

[19]This problem also goes under the name of the *curse of dimensionality* - the task that is easy in a single dimension becomes extremely complex at higher dimensions.

Rejection sampling also assumes that you have a constant M such that $\widehat{q}(\mathbf{x})$ scaled by the constant yields an upper bound for the original distribution $\wp(\mathbf{x})$.

$$\wp(\mathbf{x}) \leq M\widehat{q}(\mathbf{x})$$

The way rejection sampling works is:

1. First generate a sample $\mathbf{y} \sim q(\mathbf{x})$.
2. Secondly, sample u from a uniform distribution between 0 and $M\widehat{q}(\mathbf{y})$: $u \sim U[0, M\widehat{q}(\mathbf{y})]$.
    - If $u < \wp(\mathbf{y})$, then accept $\mathbf{y}$ as a realization of $\wp(\mathbf{x})$.
    - Otherwise, reject $\mathbf{y}$.

We will see that this random procedure itself induces a distribution p_{rej} over $\mathbf{x}$ that happens to be the same as $\wp(\mathbf{x})$. For any $\mathbf{y}$ that is an output of the rejection sampling procedure, its probability of being generated by the procedure is the product of the probability $q(\mathbf{y})$ of choosing $\mathbf{y}$ and the probability $\frac{\wp(\mathbf{y})}{M\widehat{q}(\mathbf{y})}$ of accepting $\mathbf{y}$:

$$p_{gen}^{rej}(\mathbf{y}) = \frac{1}{z_{p_{gen}^{rej}}} q(\mathbf{y}) \left(\frac{\wp(\mathbf{y})}{M\widehat{q}(\mathbf{y})} \right) = \frac{1}{z_{p_{gen}^{rej}} z_q} \wp(\mathbf{y})$$

where, the normalization constant $\frac{1}{z_{p_{gen}^{rej}}}$ is defined as

$$z_{p_{gen}^{rej}} = \int_{\mathbf{y}'} q(\mathbf{y}') \left(\frac{\wp(\mathbf{y}')}{M\widehat{q}(\mathbf{y}')} \right) d\mathbf{y}' = \int_{\mathbf{y}'} \frac{1}{z_q} \wp(\mathbf{y}') d\mathbf{y}' = \frac{z_p}{z_q}$$

Combined, these two equalities mean that

$$p_{gen}^{rej}(\mathbf{y}) = \frac{1}{z_{p_{gen}^{rej}} z_q} \wp(\mathbf{y}) = \frac{z_q}{z_{p_{gen}^{rej}} z_q z_p} \wp(\mathbf{y}) = \frac{1}{z_p} \wp(\mathbf{y}) = p(\mathbf{y})$$

In practice, it crucially matters how small you can make the reject region, since you would not like to spend too many sampling cycles to generate each sample $\mathbf{y}$. So it will be best to choose a $\widehat{q}$ that follows $\wp$ very closely. A measure of this 'following closely' is the ratios of the area A_{acc} under $\wp(\mathbf{x})$ to the area A_{tot} under $M\widehat{q}(\mathbf{x})$. This can be thought of as the acceptance probability p_{acc}^{rej}.

$$p_{acc}^{rej} = \frac{\displaystyle\int_{\mathbf{x}} \wp(\mathbf{x}) d\mathbf{x}}{\displaystyle\int_{\mathbf{x}} M\widehat{q}(\mathbf{x}) d\mathbf{x}} = \frac{A_{acc}}{M z_q} = \frac{A_{acc}}{A_{tot}}$$

One of the concerns with rejection sampling in high dimensions is that since $p_{acc}^{rej} \propto \frac{1}{M}$, the number of attempts before getting a single sample will scale as M. For instance, suppose $\mathbf{X}$ and $\mathbf{Y}$ are independent Gaussians with slightly different variances - $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}, \mathbf{0}, \sigma_p^2 I)$ and $q(\mathbf{y}) = \mathcal{N}(\mathbf{y}, \mathbf{0}, \sigma_q^2 I)$. where $\rho \in (0, 1.1]$. For rejection sampling, we will need to choose an $M > \frac{\wp(\mathbf{x})}{\widehat{q}(\mathbf{x})}$ for every $\mathbf{x} \in \Re^n$. For this Gaussian, we will require that $M > \frac{\wp(\mathbf{0})}{\widehat{q}(\mathbf{0})} = \exp\left\{n \log \frac{\sigma_q}{\sigma_p}\right\}$. Note that if σ_q is even slightly larger than σ_p, then M will increase exponentially with n. That is, we will have to do exponentially many rejection trials before getting a sample accepted.

In summary, while rejection sampling is useful in low dimensions, it breaks down in higher dimensions. Markov chain monte carlo (MCMC) builds on rejection sampling. While rejection sampling in memoriless - in the sense that rejected samples are naively abandoned, MCMC preserves rejected samples using memory.

Importance Sampling

Importance sampling is a more general form of Monte Carlo sampling method. Recall that Monte Carlo sampling was to solve the problem

$$E[f] = \int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} \tag{1.38}$$

and the Monte Carlo estimate collects iid samples $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)})$ from $p(.)$ and computes the weighted sum

$$\widehat{f} = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}^{(i)})$$

The idea of importance sampling is to generate samples $\overline{\mathbf{y}}$ from an alternative q which is somewhat easier to sample than p. However, how do we make use of $\overline{\mathbf{y}}$ in estimation? Assuming that $q(\mathbf{x}) > 0$ whenever $f(\mathbf{x})p(\mathbf{x}) > 0$, we rewrite the expression for $E[f]$ as

$$E[f] = \int \frac{p(\mathbf{x}) f(\mathbf{x})}{q(\mathbf{x})} q(\mathbf{x}) d\mathbf{x}$$

Defining $g(\mathbf{x}) = \frac{p(\mathbf{x}) f(\mathbf{x})}{q(\mathbf{x})}$, we just re-express

$$E_p[f] = E_q[g]$$

This motivates the study of the Monte Carlo estimate $\widehat{g}$ based on samples $\overline{\mathbf{y}}$ from q.

$$\widehat{g} = \frac{1}{m} \sum_{i=1}^{m} g(\mathbf{y}^{(i)}) = \frac{1}{m} \sum_{i=1}^{m} \frac{p(\mathbf{y}^{(i)}) f(\mathbf{y}^{(i)})}{q(\mathbf{y}^{(i)})} \tag{1.39}$$

The estimate $\widehat{g}$ is called the importance sampling estimate. The terms $\frac{p(\mathbf{y}^{(i)})}{q(\mathbf{y}^{(i)})}$ are called *importance weights*. It can be shown that $\widehat{g}$ is unbiased, that is,

$$E_p[f] = E_{\overline{\mathbf{y}}}[\widehat{g}]$$

Like for the case of the Monte Carlo estimate, it can also be shown that if g has finite variance then,

$$var(\widehat{g}) = \sum_{i=1}^{m} \frac{var\left(g(\mathbf{y}^{(i)})\right)}{m} \tag{1.40}$$

or equivalently, the spread for $var(\widehat{g})$ is $\frac{1}{\sqrt{m}}$ times the spread for $\widehat{g}$.

Importance sampling is useful when q is easier to sample than p. Backing off a bit, it may happen that q is itself of the form

$$q(\mathbf{y}) = \frac{1}{z_q}\widehat{q}(\mathbf{y})$$

where $\widehat{q}(\mathbf{y})$ is easy to compute while the normalization constant z_q is not. This is especially true in the case of graphical models, for which $\widehat{q}(\mathbf{y})$ is simply the product of some compatibility functions or exponentiated weighted sum of sufficient statistics (exponential family). Similarly, it is often that $p(\mathbf{x}) = \frac{1}{z_p}\wp(\mathbf{x})$. In such a case, we can write

$$\widehat{g} = \frac{1}{m}\sum_{i=1}^{m} \frac{p(\mathbf{y}^{(i)})f(\mathbf{y}^{(i)})}{q(\mathbf{y}^{(i)})} = \frac{1}{m}\sum_{i=1}^{m} \frac{\wp(\mathbf{y}^{(i)})f(\mathbf{y}^{(i)})}{\widehat{q}(\mathbf{y}^{(i)})}\frac{z_q}{z_p}$$

For this formulation, $\frac{\wp(\mathbf{y}^{(i)})}{\widehat{q}(\mathbf{y}^{(i)})}$ are called the importance weights $im(\mathbf{y}^{(i)})$. Also,

$$z_{p/q} = \frac{z_p}{z_q} = \frac{1}{z_q}\int \wp(\mathbf{x})d\mathbf{x} = \int \wp(\mathbf{x})\frac{q(\mathbf{x})}{\widehat{q}(\mathbf{x})} = \int \frac{\wp(\mathbf{x})}{\widehat{q}(\mathbf{x})}q(\mathbf{x})d\mathbf{x}$$

which is again the expectation under q of $im(\mathbf{y})$. The Monte Carlo estimate $\widehat{z}_{p/q}$ for $z_{p/q} = \frac{z_p}{z_q}$ is

$$\widehat{z}_{p/q} = \frac{1}{m}\sum_{i=1}^{m} im(\mathbf{y}^{(i)}) = \frac{1}{m}\sum_{i=1}^{m}\frac{\wp(\mathbf{y}^{(i)})}{\widehat{q}(\mathbf{y}^{(i)})}$$

This gives you a modified Monte Carlo estimate, which can be contrasted against (1.39).

$$\widehat{g}' = \frac{\frac{1}{m}\sum_{i=1}^{m}\frac{p(\mathbf{y}^{(i)})f(\mathbf{y}^{(i)})}{q(\mathbf{y}^{(i)})}}{\frac{1}{m}\sum_{i=1}^{m}\frac{\wp(\mathbf{y}^{(i)})}{\widehat{q}(\mathbf{y}^{(i)})}} = \frac{\sum_{i=1}^{m} f(\mathbf{y}^{(i)})im(\mathbf{y}^{(i)})}{\sum_{i=1}^{m} im(\mathbf{y}^{(i)}} \tag{1.41}$$

This modified Monte Carlo estimate uses samples $\overline{\mathbf{y}}$ from q and also does not require explict computation of p. Rather it needs to compute only $\wp$.

Importance sampling considerably widens the scope of Monte Carlo methods, since it provides flexibility of choosing q. In fact, even if you could sample from p, it can be useful to use a q, especially when f lies in the tail region(s) of p. Since q lets you reshape what you sample from, the modified g can help shift the mass over to regions of p such that getting information from f becomes much more likely. In practice, adapting q to the shape of f can also lead to reduction in the variance of $\widehat{g}$.

Finally, importance sampling can be useful even when you can draw samples from $p(.)$. For example, say $X_i \in \{0,1\}$, for $1 \leq i \leq n$ are n binary random variables with $p(X_i = 1) = \epsilon$ for a very small ϵ, close to 0. Let the X_i's be independent (though the example could hold for many non-independent X_i's as well). Let us say, we are interested in estimating

$$p\left(\frac{\sum\limits_{i=1}^{n} x_i}{n}\right) \geq 0.5$$

As can be seen, that $\frac{\sum\limits_{i=1}^{n} x_i}{n} \geq 0.5$ is a very rare event. Importance sampling can be used to model such rare events which are otherwise computationally infeasible to simulate.

Like in the case of rejection sampling and numerical techniques, there are problems that importance sampling techniques have in high dimensional spaces.

Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are a suite of methods based on setting up a markov chain to generate samples from a target distribution

$$p(\mathbf{x}) = \frac{1}{z}\wp(\mathbf{x})$$

where z may not be easy to compute. The basic idea here is to set up a Markov chain $X_1 \rightarrow X_2 \rightarrow \ldots X_n$ and a sequence of distributions q so that as $t \rightarrow \infty$, $q(.|X^{(t)}) \rightarrow p$. While these methods have a flavour of rejection sampling, they are not memoryless; rejected samples are not entirely discarded. Introduced by physicists Metropolis, Rosenbluth, Teller in 1953, these methods were generalized by Hastings in 1970.

The Metropolis-Hastings algorithm is one of the more popular examples from this class. It is outlined in Figure 1.12. It builds on the rejection sampling technique, with two main differences. More importantly, a sample is not discarded if rejected; the value of X_{t+1} is set to X_t. Secondly, the acceptance probability determines if it is more likely to go $\mathbf{X}^{(t)} \rightarrow \mathbf{y}$ or $\mathbf{y} \rightarrow \mathbf{X}^{(t)}$.

```
Input: A target distribution $p(\mathbf{x}) = \frac{1}{z}\wp(\mathbf{x})$.
Initialize: $\mathbf{X}^{(1)} \sim q(\mathbf{x})$ for some arbitrary $q$.
for $t = 1, 2, \ldots$ do
    1. Sample $\mathbf{y}$ from the conditional probability distribution $q(.|\mathbf{X}^{(t)})$.
    2. Sample $u \sim Uniform[0,1]$.
    3. Define the acceptance probability as
       $$A(\mathbf{X}^{(t)}, \mathbf{y}) = \min \left\{1, \frac{\wp(\mathbf{y})q(\mathbf{X}^{(t)}|\mathbf{y})}{\wp(\mathbf{X}^{(t)})q(\mathbf{y}|\mathbf{X}^{(t)})}\right\}$$
    4. Set
       $$\mathbf{X}^{(t+1)} = \begin{cases} \mathbf{y}, & \text{if } u \le A(\mathbf{X}^{(t)}, \mathbf{y}) \quad //\text{Accept} \\ \mathbf{X}^{(t)} & \text{otherwise} \quad //\text{Reject, but do not discard.} \end{cases}$$
end for
```

Figure 1.12: The Metropolis-Hastings algorithm.

In the special case of symmetry, that is if $q(\mathbf{y}|\mathbf{X}^{(t)}) = q(\mathbf{X}^{(t)}|\mathbf{y})$,

$$A(\mathbf{X}^{(t)}, \mathbf{y}) = \min \left\{1, \frac{\wp(\mathbf{y})}{\wp(\mathbf{X}^{(t)})}\right\}$$

the algorithm acts like a gradient algorithm with some randomness. To see this, note that if $\wp(\mathbf{y}) \geq \wp(\mathbf{X}^{(t)})$, the algorithm will always accept. Else it accepts with probability $\frac{\wp(\mathbf{y})}{\wp(\mathbf{X}^{(t)}} < 1$; the logic is that you do not always want to reject even if you were to go downhills, since you might want to wriggle out of local modes. So you reject, but probabilistically. Thus, the algorithm always tries to sample from regions of the space where the density is more.

We will eventually see that if $q(\mathbf{y}|\mathbf{X}^{(t)}) \to p(\mathbf{y})$ as $t \to \infty$. Enroute to proving this, we will require to understand the limiting behaviour of Markov chains as number of states goes to ∞.

1. The Metropolis-Hastings algorithm in Figure 1.12 generates a first order Makov chain. Assume for simplicity that the Markov chain is finite state and that $\mathbf{X}^{(i)} \in \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$. For many graphical models, k could be exponential in the number of nodes in the graphical model. For the Markov chain $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \mathbf{X}^{(3)}, \ldots$ generated by the algorithm in Figure 1.12, $\mathbf{X}^{(1)} \sim q(.)$ while the transition $\mathbf{X}^{(t)} \to \mathbf{X}^{(t+1)}$ is specified by the homogeneous (*i.e.* fixed across time steps) conditional distribution

$$\Gamma(\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}) = A(\mathbf{X}^{(t)}, \mathbf{y})q(\mathbf{y}|\mathbf{X}^{(t)})$$

That is, the transition function for the algorithm is the combination of the original proposal distribution and the probability of acceptance in the accept/reject step. Then, by the steps (1) and (4) of the algorithm in Figure 1.12,

$$q(\mathbf{X}^{(t+1)}) = \sum_{\mathbf{X}^{(t)}} q(\mathbf{X}^{(t)})\Gamma(\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)})$$

In matrix notation, this translates to

$$\mathbf{q}_{t+1} = \mathbf{q}_t^T T \tag{1.42}$$

where, $\Gamma[i,j] = \Gamma(\mathbf{X}^{(t+1)} = \mathbf{x}_j|\mathbf{X}^{(t)} = \mathbf{x}_i)$ and $\mathbf{q}_t[i] = q(\mathbf{X}^{(t)} = \mathbf{x}_i)$. By its very definition, Γ is row-stochastic, that is,

$$\Gamma\mathbf{1} = \mathbf{1}$$

We would like $\mathbf{q} \rightarrow \mathbf{p}$, where $\mathbf{p}$ is the targetted probability vector. The following sequence of arguments will help arrive at conditions under which this will happen.

2. Let $\mathbf{r}$ be the fix point of update equation (1.42). Then $\mathbf{r}$ is also invariant[20] with respect to Γ. Thus, once the distribution $\mathbf{q}_t$ hits an invariant $\mathbf{r}$, it stays in $\mathbf{q}_t$.

3. In order to have an invariant vector, the matrix must be non-negative ($\Gamma \geq 0$) and must be row-stochastic, which it is. The matrix Γ is not symmetric in general. The Perroon Forbenius theorem states that for any non-negative, row-stochastic matrix A, its spectral radius $\rho(A) = \max\limits_{i=1,2,\ldots,n} |\lambda_i(A)|$ satisfies the condition $\rho(A) = 1$ and that it has a left eigenvector $\mathbf{v} \geq \mathbf{0}$ such that $\mathbf{v}^T A = \mathbf{v}$. Since the matrix Γ is both non-negative and row-stochastic, a consequence of this theorem is that $\mathbf{r} = \frac{1}{\sum\limits_{i=1}^{k} v_k}\mathbf{v}$ will be invariant with respect to Γ.

4. The above conditions and arguments state in principle that there is potentially a fix point for (1.42). In general, the fix point need not be unique; for a diagonal matrix, there are several invariant distributions. It can shown that an irreducible matrix Γ (*i.e.*, every state is reachable from every other state with strictly positive probability) will have a unique invariant distribution $\mathbf{r}$.

[20] A vector $\mathbf{r} \in \Re^k$ is invariant with respect to matrix Γ if $\mathbf{r}$ represents a probability distribution (*i.e.*, $\mathbf{r}^T\mathbf{1} = \mathbf{1}$ and $\mathbf{r} \geq \mathbf{0}$) and is fixed under the updates of Γ, that is $\mathbf{r}^T\Gamma = \mathbf{r}$. As an example, you can verify that for random walks on a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with 0 jump probability, $\mathbf{r}$ such that $v_i = \frac{d_i}{\sum\limits_{i\in\mathcal{V}} d_i}$ is invariant, d_i being the degree of vertex i.

5. But will the algorithm in Figure 1.12 converge to the fix point? To enable convergence, another necessary condition is that the Markov chain should be aperiodic (so that q_t does not keep toggling between values) or equivalently, have a period of 1.

6. With all the armour discussed so far, we state the following theorem, central to the correct convergence of the algorithm in Figure 1.12:

   Theorem 12 *For any finite-chain irreducible aperiodic Markov Chain, the sequence* $\mathbf{q}_{t+1} = \mathbf{q}_t^T \Gamma$ *converges, for an arbitrary* $\mathbf{q}_1$*, to the unique invariant distribution* $\mathbf{r}$ *of the chain.*

   The next few steps are dedicated to proving that (i) the Metropolis-Hastings algorithm satisfies the pre-requisites for this theorem under certain conditions and (ii) that the target distribution is indeed invariant with respect to the Markov chain in the Metropolis-Hastings algorithm.

7. The next step is to establish that the matrix Γ defined as

   $$\Gamma(\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}) = A(\mathbf{X}^{(t)}, \mathbf{y})q(\mathbf{y}|\mathbf{X}^{(t)})$$

   is irreducible and aperiodic. The Metropolis-Hastings algorithm is very general, allowing fairly arbitrary proposal distribution. Both these properties can be established if $q(\mathbf{y}|\mathbf{X}^{(t)}) > 0$ for all $\mathbf{y}$, so that $\Gamma(\mathbf{X}^{(t+1)}|\mathbf{X}^{(t)}) > 0$ for all $\mathbf{y}$. For complex models such as graphical models, the q should be designed so as to make that task of sampling from q efficient. Gibbs sampling is one such example.

8. The final step is in showing that the target distribution $\mathbf{p}$ is invariant for the Markov chain Γ. That is, for every $\mathbf{y}$,

   $$\sum_{\mathbf{x}} p(\mathbf{x})\Gamma(\mathbf{y}|\mathbf{x}) = p(\mathbf{y})$$

   This can be shown by first noting that the Metropolis Hastings Markov chain Γ satisfies the *detailed balance condition* with respect to $\mathbf{p}$, which means

   $$p(\mathbf{x})\Gamma(\mathbf{y}|\mathbf{x}) = p(\mathbf{y})\Gamma(\mathbf{x}|\mathbf{y})$$

   This can be proved by simply substituting for $\Gamma(\mathbf{y}|\mathbf{x})$. This leads to the desired result

   $$\sum_{\mathbf{x}} p(\mathbf{x})\Gamma(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{x}} p(\mathbf{y})\Gamma(\mathbf{x}|\mathbf{y}) = p(\mathbf{y})$$

   since Γ is row-stochastic.

While the algorithm in Figure 1.12 works in principle, it can be very slow, taking small steps into valleys. In practice, many refinements are made to the algorithm to make it work fast.

Gibbs Sampling

Gibbs sampling is a simple subclass of Metropolis-Hastings algorithm in which the transitions Γ are intuitive and easy to draw from. It is especially relevant for graphical models, which have a large state space of size c^n if each random variable can take c values and the graph has n nodes. Let $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ be a graph with n nodes. The Gibbs sampling procedure involves two main steps, as outlined in Figure 1.13 It is very natural to think of Gibbs sampling in terms of

```
**Input**: A target distribution $p(\mathbf{x}) = \frac{1}{z}\wp(\mathbf{x})$ over a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$.
**Initialize**: $\mathbf{X}^{(1)} = \mathbf{x}^{(1)}$ for some arbitrary $q$.
**for** step $t = 1, 2, \ldots$ **do**
  **for** $i = 1, \ldots, n$ **do**
    Sample: $x_i^{(t+1)} \sim p(x_i \mid \mathbf{x}_{1,i-1}^{(t+1)} \cup \mathbf{x}_{i+1,n}^{(t)}$.
  **end for**
**end for**
```

Figure 1.13: The Gibbs sampling algorithm.

the graphical model; the sampling is very much simplified given the conditional independence property - that a node is indepent of all other nodes, given its Markov blanket. Thus, each distribution in the sampling step is the probability of a node given its Markov blanket. This procedure depends on the ordering of nodes, though. Though a very popular inference procedure for graphical models, it could take an extemely long time to converge.

For discrete random variables, the graph of configurations is in general a hypercube. While MCMC potentially allows jumps from one node to another in the hypercube, Gibbs sampling restricts every step to be along an edge of the hypercube. This can lead to poor convergence. There are many smarter algorithms that take larger, but calculated steps in the hypercube, leading to better convergence, without incurring excessive computational cost for individual steps.

1.3 Factor Graphs

We have seen that both directed and undirected models can be specified in one of two equivalent ways each; conditional independence and factorization. The semantics for directed and undirected models are however different. While the two specifications are equivalent, factorization can be specified at different levels of granularity or could be defined using different forms of potential functions and is therefore richer. For instance, factorization could be over maximal cliques or over smaller cliques or even over edges, while all these specifications could map to the same conditional independence properties demanding specialization in the conditional independence assertions.

Consider a triangular graph, given by the adjacency matrix

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The factorization for this graph could be presented as

$$p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_2, x_3)$$

However, the following factorization is also a valid one for the graph

$$p(x_1, x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{23}(x_2, x_3)\phi_{13}(x_1, x_3)$$

Such problems interest statisticians a lot, since existence of interactions between variables influence diagnosis and/or treatments in the medical domain, for example. Is it possible to graphically differentiate between the two factorizations? Factor graphs precisely serve this purpose. In some sense, there is more information being embedded in the second factorization; the bonafide triplet interaction has been decomposed into pairwise interactions. And this information is represented as 'factor nodes' in factor graphs. Corresponding to each factor in the factorization, factor graphs host a node. The nodes in the original graphical model are also retained. But edges are changed; there will be an edge between the factor node and each node whose random variable the factor is a function of.

Definition 13 *Given a graphical model $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, with a factorization $p(\mathbf{x}) = \prod_{a \in \mathcal{F}} \phi_a(\mathbf{x}_a)$ with $\mathcal{F} \subseteq 2^{\mathcal{V}}$, that is compatible with the independence assumptions asserted by the edges $\mathcal{E}$, the factor graph $\mathcal{G}_f =< \mathcal{V} \cup \mathcal{F}, \mathcal{E}_f >$ is defined by $\mathcal{E}_f = \{(q, a) \mid q \in \mathcal{V}, a \in \mathcal{F}, q \in a\}$ The factor graph $\mathcal{G}_f$ is said to encode the factorization of $\mathcal{G}$.*

The set of factor nodes $\mathcal{F}$ is a set of placeholders for particular terms in the factorization. Factor graphs can be looked upon as a 'graphical way' of representing hypergraphs (which can have a single edge spanning multiple vertices), where each hyperedge can be thought of as spanning all vertices that figure together in some potential function in the factorization.

As an example, the factor graph for $p(x_1, x_2, x_3) \propto \phi_{123}(x_1, x_2, x_3)$ will be given by $\mathcal{V} = \{1, 2, 3\}$, $\mathcal{F} = \{a\}$ and $\mathcal{E}_f = \{(1, a), (2, a), (3, a)\}$. Whereas, the factor graph for $p(x_1, x_2, x_3) \propto \phi_{12}(x_1, x_2)\phi_{13}(x_1, x_3)\phi_{23}(x_2, x_3)$ will be given by $\mathcal{V} = \{1, 2, 3\}$, $\mathcal{F} = \{a, b, c\}$ and $\mathcal{E}_f = \{(1, a), (2, a), (1, b), (3, b), (2, c), (3, c)\}$.

Any undirected/directed graph without any specialized factorization specified can also be converted into a factor graph. Thus, if $\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7, 8\}$, $\mathcal{E} = \{(1, 2), (2, 3), (3, 4), (2, 5), (3, 4), (4, 5), (4, 7), (5, 7), (5, 6), (6, 8), (7, 8)\}$, then $\mathcal{G}_f$ can be read off the maximal cliques; $\mathcal{F} = \{a, b, c, d, e\}$ and $\mathcal{E}_f = \{(1, a), (2, a), (3, a), (2, b), (5, b), (3, c), ($
As another example, consider a hidden markov model with $\mathcal{V} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$

and $\mathcal{E} = \{(1,2),(1,6),(2,3),(2,7),(3,4),(3,8),(4,5),(4,9),(5,10)\}$. Then the factor graph will have $\mathcal{F} = \{a,b,c,d,p,q,r,s\}$ and $\mathcal{E}_f = \{(1,a),(2,a),(2,b),(3,b),(3,c),(4,c),(4,d),(5,d),(1,p),(6,p),$
As a final example, consider a markov decision process, which is a purely directed model and which has five 'control' variables in addition to the 10 for the HMM described above (decision process because there is a control that helps you decide the behaviour of the evolution across states in the HMM). It will be specified by $\mathcal{V} = \{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15\}$ and $\mathcal{E} = \{(1,2),(1,6),(11,1),(2,3),(2,7),(12,2),(3,4),(3,8),(13,3),(4,5),(4,9),(14,4),(5,10),(15,5)\}$. What will be the factor graph corresponding to this graph? Note that, even though there are no directed triangles in this graph, nodes 1 throug 5 have two parents each and the corresponding factors are conditional probabilities of the form $p(x_1|x_6,x_{11})$, $p(x_2|x_7,x_{12})$, *etc.* Thus, the factor graph will have a node for each such factor connected to three nodes each.

All the factor graph examples considered thus far are factor trees. The sum-product and max-prodyct algorithms are exact on factor trees. Though they assume slightly different forms, the essential ideas are the same. There is no consistent reason for the factor graph being a better representation than the graphical model representation.

1.4 Exponential Family

The exponential family captures many common discrete[21] and continuous[22] graphical model formulations at an abstract level. It provides a rich (though not exhaustive) toolbox of models. The multinomial distribution which models random variables taking k discrete values is what we have looked at so far. Gaussian is one of the most widely used continuous distributions. The poisson distribution helps model distribution over random variables that can take any integral value (as against just k discrete values).

Definition 14 *For a given vector of functions $\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}),\ldots,f_k(\mathbf{x})]$ and a parameter vector $\eta \in \Re^k$, the exponential family of distributions is defined as*

$$p(\mathbf{x},\eta) = h(\mathbf{x})\exp\left\{\eta^T\mathbf{f}(\mathbf{x}) - A(\eta)\right\} \tag{1.43}$$

where the $h(\mathbf{x})$ is a conventional reference function and $A(\eta)$ is the log normalization constant[23] designed as

$$A(\eta) = \log\left[\int_{\mathbf{x}\in Range(\mathbf{X})} \exp\{\eta^T\mathbf{f}(\mathbf{x})\}h(\mathbf{x})d\mathbf{x}\right]$$

The domain of the parameters η will be restricted to $Domain(\eta) = \left\{\eta \in \Re^k \mid A(\eta) < +\infty\right\}$. $A(\eta)$ plays a fundamental role in hypothesis testing, parameter estimation, etc.,

[21]Density with respect to the Counting measure.
[22]Density with respect to the Lebesgue measure.
[23]The same as $\log Z$ for graphical models.

though often not very easy to estimate. The central component here is the log-linear form. The parametrization $\eta \in \Re^k$, is also called the canonical parametrization. The function $\mathbf{f}(\mathbf{x})$ is a sufficient statistic function.

As an example, the Gaussian density function can be expressed in the exponential form. If $X \sim \mathcal{N}(\mu, \sigma^2)$ is a univariate Gaussian distribution, then its normal parametrization is

$$p(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

This density can be manipulated and shown to be a member of the exponential family

$$p(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}} \exp\left\{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \log\sigma\right\} = p(x, \eta)$$

The parameter vector for the exponential form is $\eta = \left[\frac{\mu}{\sigma^2}, -\frac{1}{2\sigma^2}\right]$ while the feature function vector is $\mathbf{f}(x) = [x, x^2]$. The log normalization constant is $A(\eta) = \frac{1}{2\sigma^2}\mu^2 + \log\sigma \equiv -\frac{\eta_1^2}{4\eta_2} + \frac{1}{2}\log(-2\eta_2)$, which determines $Domain(\eta) = \left\{\eta \in \Re^2 \mid \eta_2 < 0\right\}$. Finally $h(\mathbf{x}) = \frac{1}{\sqrt{2\pi}}$ for this example. The number of degrees of freedom (reflected through two parameters in the moment parametrization) will precisely be the number of canonical parameters. The canonical parametrization extended[24] to the multivariate Gaussian counterpart will be discussed in Section 1.5.

As a second example, consider the bernoulli distribution. A bernoulli random variable $X \in \{0, 1\}$ can model the outcome of any coin-flipping experiment. It can be expressed as

$$p(x, \mu) = \mu^x(1-\mu)^{1-x}$$

where μ is the probability of $X = 1$. Rearranging the terms, we get the exponential form for bernoulli distribution as

$$p(x, \mu) = \exp\left\{\left(\log\frac{\mu}{1-\mu}\right)x + \log(1-\mu)\right\} = p(x, \eta)$$

with parameter vector specified as $\eta = \left[\log\frac{\mu}{1-\mu}\right]$ which gives μ as a logistic function (log of likelihood ratio or log-odds ratio) of η_1 and $\mathbf{f}(x) = [x]$. The log normalization constant is $A(\eta) = -\log(1-\mu) \equiv \log\{1 + e^{\eta_1}\}$[25] while $h(x) = 1$. Also, $Domain(\eta) = \Re$. In general, if you started coupling together multiple distributions and try expressing them in exponential form, determining η could become a very hard problem.

As a third example, consider the poisson[26] random variable $X \in \mathcal{N}$ (set of natural numbers). Its density function is given by

$$p(x, \mu) = \frac{\mu^x e^{-\mu}}{x!}$$

[24]EXERCISE.

[25]EXERCISE.

[26]When London was being bombed in World war 2, experts were trying to model where the bomb would next fall using a 2D poisson distribution.

Poisson distributions are often used to model events, such as the ringing of a telephone. The mean parameter μ controls the density or frequency with which the event is happening. The canonical parametrization for this distribution can be obtained using a routine rewrite as

$$p(x,\mu) = \frac{1}{x!}\exp\{(\log\mu)x - \mu\}$$

where $h(x) = \frac{1}{x!}$, $\eta = [\log\mu]$, $\mathbf{f}(x) = [x]$, $A(\eta) = \mu = e^{\eta_1}$ with $Domain(\eta) = \Re$.

In all examples considered so far, we algebriacally converted the density function from a moment parametrization to a canonical representation as a member of the exponential family. How about going backwards from a canonical form to moment parametrization? The vector of moment parameters is defined as $\mu = [\mu_1, \mu_2, \ldots \mu_k] = [E[f_1(\mathbf{X})], E[f_2(\mathbf{X})], \ldots, E[f_k(\mathbf{X})]]$. That is, we can get the moment parameters by taking expectations of the sufficient statistics $f_i(\mathbf{X})$ that sit in the exponent of the canonical form. And the expecation of the i^{th} component of this function can be proved to be the same as $\frac{\partial A}{\partial \eta_i}$. That is

$$\nabla A(\eta)_i = \frac{\partial A}{\partial \eta_i} = E[\mathbf{f}(\mathbf{X})] = \mu_i \tag{1.44}$$

Further, it can be proved that

$$\nabla^2 A(\eta)_{ij} = \frac{\partial^2 A}{\partial \eta_i \partial \eta_j} = cov\{f_i(\mathbf{X}), f_j(\mathbf{X})\} = E\left[(f_i(\mathbf{X}) - \mu_i)(f_j(\mathbf{X}) - \mu_j)\right] = \mu_{ij}$$

The proof of the two above statements are straightforward and use the property that $A(\eta)$ is infinitely differentiable. To illustrate this, we will revisit the canonical parametrization for the Gaussian example.

$$p(x,\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}}\exp\left\{\frac{\mu}{\sigma^2}x - \frac{1}{2\sigma^2}x^2 - \frac{1}{2\sigma^2}\mu^2 - \log\sigma\right\} = p(x,\eta)$$

The moments can be derived to me $\mu_1 = E[x] = \mu$ and $\mu_2 = E[x^2] = \sigma^2 + \mu$. Similarly, for the poisson distribution, the moment parameter is simply $[\mu]$ as also $var(X) = \mu$. For the bernoulli distribution, $E(X) = \mu$ and $var(X) = (1-\mu)\mu$.

1.5 A Look at Modeling Continuous Random Variables

A normal or gaussian distribution is one of the most widely (ab)used probability distributions. They come up in the central limit theorem in the sums of independent or weakly dependent random variables, whose normalized sum coverges to a gaussian (justifying their wide use). They are very often used to model noise. Example graphical models that use gaussian distribution are Gaussian graphical models, Gauss-Markov time series, *etc.*

Another reason for their wide use is computational. In the case of continuous random variables, messages are functions rather than vectors. In general, this can often force us to quantize the messages. But in the case of gaussians (and in general for exponential models which we will shortly see), the message passing can be performed in terms of sufficient statistics. Kalman filters are a special case of message passing that pass sufficient statistics as messages.

The form of the density function for Gaussian distribution, with $\mathbf{x}, \mu \in \Re^n$ and $\Sigma \in \Re^{n \times n}$ is

$$p(\mathbf{x}, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{1}{2}}\sqrt{det(\Sigma)}} \exp\left\{\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right\} \tag{1.45}$$

where μ is the mean vector and $\Sigma \succ \mathbf{0}$ is the covariance matrix[27]. It can be verified that μ is indeed the mean vector; $\mu = E[\mathbf{X}] = \int_{\Re^n} \mathbf{x}p(\mathbf{x}, \mu, \Sigma)d\mathbf{x}$. Similarly, it can be verified that $\Sigma = E[(\mathbf{X}-\mu)(\mathbf{X}-\mu)^T]$. The quantity $\frac{1}{(2\pi)^{\frac{1}{2}}\sqrt{det(\Sigma)}}$ is the normalization constant and can be computed as $\int_{\Re^n} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right\}d\mathbf{x}$. If $n = 1$, the integral is easy to compute. For computing integrals for multivariate problem, it is a good idea to reduce the matrix Σ by diagonalization.

What sits in the exponent of the density function is a quandratic term. The contours of constant probability are ellipsoids, with shape determined by Σ^{-1} and center as μ (which does not affect the shape of the ellipsoid). For example, if Σ^{-1} is diagonal, the axes of the ellipsoid will be aligned along the coordinate axes.

The parametrization (μ, Σ) is called the *moment parametrization* of the Gaussian; μ is the first order moment of $\mathbf{x}$ while Σ is the matrix of second order centered moment. There is another *canonical parametrization* of the Gaussian which is related to the sum-product algorithm. The parameters are a new vector and a new matrix:

$$\eta = \Sigma^{-1}\mu$$

and a new matrix

$$\Omega = \Sigma^{-1}$$

With a bit of algebra, the Gaussian density can be re-expressed in terms of the canonical parameters as:

$$p(\mathbf{x}, \eta, \Omega) = \exp\left\{\eta^T\mathbf{x} - \frac{1}{2}\mathbf{x}^T\Omega\mathbf{x} + \mathbf{x}\right\} \tag{1.46}$$

[27] Recall that a matrix is positive definite if all its eigenvalues are strictly positive or equivalently, $\forall \mathbf{z} \neq \mathbf{0},\ \mathbf{z}^T\Sigma\mathbf{x} > 0$.

where

$$\mathbf{x} = -\frac{1}{2}\left\{n\log(2\pi) - \log|\Omega| + \eta^T\Omega^{-1}\eta\right\}$$

is a constant, analogous to the normalization constant in the moment parametrization of the Gaussian density in (1.45). The parametrization for (1.45) is more naturally suited for graphical models, because, as we will see, the canonical parameters can be directly related to the compatibility functions when you start breaking up the Gaussian into a graphical model-like factorization. Typically, graphical model factorizations invoke canonical parameters. The above parametrization is often referred to as $\mathcal{N}(\eta, \Omega)$.

Equation (1.46 gives a quadratic form in the exponent and is a special case of the exponential family representation, which happens to be a very general concept. Note that all logarithms are to the base e. The quadratic term in the exponent can be rewritten as a trace:

$$\mathbf{x}^T\Omega\mathbf{x} = trace(\Omega\mathbf{x}\mathbf{x}^T)$$

where $trace(A)$ is the sum of the diagonal entries of A and happens to be linear. The Gaussian density using this transformation is obtained in its log-normal form, which has an exponent linear in its features of $\mathbf{x}$ and $\mathbf{x}\mathbf{x}^T$.

It can be shown that linear functions of Gaussian random vectors are Gaussian; $\sum\limits_{i=0}^{n} Y_i \sim \mathcal{N}(0,(n+1)\sigma^2)$ if each $Y_i \sim \mathcal{N}(0,\sigma^2)$. Also, if $X_i = \sum\limits_{j=0}^{n} Y_i$ then $p(x_{i+1} \mid x_i) = \exp\left\{-\frac{1}{2\sigma^2}(x_{i+1}-x_i)^2\right\}$ and the random variables X_i form a markov chain with factorization

$$p(\mathbf{x}) = \exp\left\{-\frac{1}{2\sigma^2}x_0^2\right\}\prod_{i=1}^{n}\exp\left\{-\frac{1}{2\sigma^2}(x_{i+1}-x_i)^2\right\}$$

We can verify that for this markov chain, $E[X_1] = E_{X_0}\left[E_{X_1|X_0}[X_1 \mid X_0]\right] = E_{X_0}\left[X_0 + E[Y_0]\right] = 0$ and in general $E[X_i] = 0$. Also, $E[X_1X_2] = 2\sigma^2$.

The canonical parameters for $p(\mathbf{x})$ can be read off the factorization as $\eta = \mathbf{0}$ and a tridiagonal, sparse

$$\Omega = \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ 0 & -1 & 2 & \dots & 0 \\ 0 & 0 & -1 & \dots & 0 \\ . & . & . & \dots & . \\ . & . & . & \dots & -1 \\ 0 & 0 & 0 & \dots & 2 \end{bmatrix}$$

The matrix Ω is sparse, because the graph is sparse (just a markov chain). Missing edges translate to zeros; for any graph, if there is no edge between X_i and X_j, $\Omega_{ij} = 0$.

Factor analysis is another example application of gaussian density functions. Principal component analysis and Karhunen Loeve transform are limiting cases of factor analysis. The motivation here is dimensionality reduction for cases when $\mathbf{Y} \in \Re^n$ and n is very large, such as the size of an image for a naive $1-d$ raster scanned pixel vector representation. The complete vector need not really capture the intrinsic dimensionality of the object being described (such as a face). Factor analysis is useful if you think that the data should be lying in some lower ($d << n$) dimensional space[28]. Let $\omega_1, \omega_2, \ldots, \omega_d \in \Re^n$ be d (linearly independent) basis vectors. Consider the linear subspace $M = \left\{\mathbf{y} \in \Re^n \middle| \mathbf{y} = \sum_{i=1}^{d} x_i\omega_i,\ x_i \in \Re,\ \omega_i \in \Re^n\right\}$. Factor analysis tries to induce a probabilitic distribution over the subspace M, by defining $\mathbf{X} \sim \mathcal{N}(\mathbf{0}_d, I_{d\times d})$ and

$$\mathcal{Y}_{n\times 1} = \mu_{n\times 1} + \Omega_{n\times d}\mathbf{X}_{d\times 1} + \mathbf{K}_{n\times 1}$$

where the i^{th} column of Ω is ω_i, $\mathbf{K} \sim \mathcal{N}(\mathbf{0}, D)$ is gaussian noise (capturing the assumption that the model may not be exactly correct) and $\mu \in \Re^d$ is the shift (in subspace $\Re^n$) vector. Though very naive, it has not stopped researchers and especially practitioners from using it successfully. The components ω_i's are extracted from a large database using eigenvalue analysis (such as eigenface analysis in image processing literature).

The graphical model representation for factor analysis is very simple; $\mathcal{V} = \{X_1, X_2, \ldots, X_d, \mathbf{Y}\}$ and $\mathcal{E} = \{(X_1, \mathbf{Y}), (X_2, \mathbf{Y}), \ldots, (X_d, \mathbf{Y})\}$. The X_i's are marginally independent as indicated by $\mathcal{G}$. Further, we can infer that $E[\mathbf{Y}] = \mu$ and

$$\Sigma = E\left[(\mathbf{Y}-\mu)(\mathbf{Y}-\mu)^T\right] = E\left[(\Omega\mathbf{X}+\mathbf{K})(\Omega\mathbf{X}+\mathbf{K})^T\right] = \Omega\Omega^T + D$$

Finally, you can show that

$$\begin{pmatrix} \mathbf{X}_{d\times 1} \\ \mathbf{Y}_{n\times 1} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{0}_{d\times 1} \\ \mu_{n\times 1} \end{pmatrix}, \begin{pmatrix} I_{d\times d} & \Omega^T_{d\times n} \\ \Omega_{n\times d} & \left(\Omega\Omega^T + D\right)_{n\times n} \end{pmatrix}\right)$$

In practice, it is useful to infer a distribution for $\mathbf{X}$ (distribution on weights for different eigenfaces) used to synthesize a particular observation $\mathbf{Y} = \widehat{\mathbf{y}}$ (such as a face image). That is, it can be required to infer $p(\mathbf{X} \mid \widehat{\mathbf{y}})$. Fortunately[29], this turns out to be a Guassian and this can be inferred using the Bayes rule and

[28]There is a lot of interest these days in identifying the manifold (especially non-linear subspaces, such as spheres) in which the data lies. In the classical technique of factor analysis, we make a very restrictive assumption that the data lies in some *linear subspace*.

[29]The conditional distribution need not always stay in the same family. But for Gaussians, the conditional distribution stays in the same family.

algebraic operations on matrices. In particular, the following property turns out to be useful. If

$$\begin{pmatrix} \mathbf{X} \\ \mathbf{Y} \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} \right)$$

then, $\mu_{\mathbf{X}|\mathbf{y}} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{y} - \mu_2)$, $Var(\mathbf{X} \mid \mathbf{y}) = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$ and $Var(\mathbf{X} \mid \mathbf{y}) \preceq \Sigma_{11} = Var(\mathbf{X})$. The last expression indicates that observation over $\mathbf{Y}$ should reduce uncertainity over $\mathbf{X}$.

1.6 Exponential Family and Graphical Models

We will now discuss the exponential family in the setting of graphical models. Particularly, we will discuss how to stitch together exponential models on graphs. Many, if not all graphical models take an exponential form. We will mainly discuss undirected graphs. Recall from (1.12), the standard factorization for an undirected graphical model

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})$$

where $\phi_{\mathcal{C}}$ is a compatibility or potential function and Π is the set of all maximal cliques. All we do in exponential form is rewrite it in the log form to get an additive decomposition. This does not always hold, since will need that all $\phi_{\mathcal{C}}$ are non-negative in order to take their logs.

$$p(\mathbf{x}) = \exp\left\{ \sum_{\mathcal{C} \in \Pi} \log \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) - \log Z \right\}$$

The above additive decomposition is essentially in exponential form, though not quite in the exponential family form (we still need to specify the canonical parameters). As an example, consider a Gaussian graphical model with $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \Omega)$ (canonical parametrization) defined on a tree $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with noisy observations $\mathcal{Y} = C\mathcal{X} + \mathcal{V}$, $\mathcal{V} \sim \mathcal{N}(\mathbf{0}, R)$. Then the conditional distribution $p(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$ is also Gaussian, which can be decomposed according to the tree structure of $\mathcal{G}$.

$$p(\mathbf{x}|\mathbf{y}) \propto \prod_{s \in \mathcal{V}} p(y_s|x_s) \prod_{(s,t) \in \mathcal{E}} \exp\left\{ -\frac{1}{2}[x_s\ x_t] J_{st} [x_s\ x_t]^T \right\}$$

where,

$$J_{st} = \begin{bmatrix} \Omega_{s(t)} & \Omega_{ts} \\ \Omega_{st} & \Omega_{t(s)} \end{bmatrix}$$

with $\Omega_{s(t)}$ specified so that $\Omega_{ss} = \sum_{t \in \mathcal{N}(s)} \Omega_{s(t)}$. In this representation, it is possible to do message passing using the canonical parameters in the sum/product updates (the basis of Kalman filters).

You could also have bernoulli random variables at every node (for example, to model the spread of a contagious disease in a social network of people) and one might be interested in building a probability distribution over the social network. Then, with every node $X_s \sim Ber(\lambda_s) \in \{0,1\}$ (canonical parametrization) and choosing[30] $\phi_{st} = \exp\{\lambda_{st} f_{st}(x_s, x_t)\}$ where, $f_{st} = x_s x_t + (1-x_s)(1-x_t)$ defined so that it takes value 1 iff $x_s = x_t$ (both are diseased or healthy) and is 0 otherwise, we have the following factorization:

$$p(\mathbf{x}) = \prod_{s \in \mathcal{V}} \exp\{\lambda_s x_s\} \prod_{(s,t) \in \mathcal{E}} \exp\{\lambda_{st} f_{st}(x_s, x_t)\} \tag{1.47}$$

If $\lambda_{st} = 0$, it indicates no coupling between the related individuals. For a reasonably contagious disease, we expect λ_{st} to be non-negative atleast. The value of θ_s indicates the belief that an individual X_s was intially diseased; higher the value, more will be the belief. Though not directly reflecting the marginal probability, λ_s is an indicator of the log-odds ratio.

Let us consider another instance - a problem of counting the vertex coverings in a graph using the sum-product formalism. We will associate a random variable $X_i \in \{0,1\}$ with each node in the graph. Then, the following exponential form for $p(\mathbf{x})$ will serve our purpose:

$$p(\mathbf{x}, \eta) = \exp\left\{\eta \sum_{i=1}^{n} x_i - A(\eta)\right\} \prod_{(s,t) \in \mathcal{E}} (1 - (1-x_s)(1-x_t))$$

where $h(\mathbf{x}) = \prod_{(s,t) \in \mathcal{E}} (1 - (1-x_s)(1-x_t))$ ensures that we indeed have a vertex cover and $\mathbf{f}(\mathbf{x}) = \sum_{i=1}^{n} x_i$. It can be proved that as $\eta \to -\infty$ (which means you are increasingly penalizing larger coverings), the distribution goes to the minimum cardinality vertex covering (*i.e.*, $\sum_{i=1}^{n} x_i$).

If a graphical model were to be used to represent a language model for spell-checking or validity, *etc.* of an input string $\mathbf{x}$ of length upto n, you can have a substantially fewer number of parameters than if you were to have a naive potential over all 26 characters in an alphabet leading to a table of size 26^n. This parameter reduction can be achieved by having indicator feature functions

[30]Remember that in a graphical model, we are free to choose the form of the potential functions so that they match the semantics. Here we pick edge potentials that reflect the coupling between health of related individuals, thereby shaping the distribution.

$\mathbf{f}(\mathbf{x})$ corresponding to interesting (and not all) substrings such as 'ion', 'ing'. To model such apriori information about 'striking patterns', it is useful to think of graphical models in the following *reduced parametrization* form, where feature function f_α can represent some such information as a feature function:

$$p(\mathbf{x}, \eta) = h(\mathbf{x}) \exp\left\{\sum_{\mathcal{C}\in\Pi} \sum_{\alpha\in I_{\mathcal{C}}} \eta_\alpha f_\alpha(\mathbf{x}_{\mathcal{C}}) - A(\eta)\right\} \tag{1.48}$$

where $I_{\mathcal{C}}$ is the index for clique $\mathcal{C}$ so that $f_\alpha(\mathbf{x}_{\mathcal{C}})$ corresponds to only those interesting features that are local in terms of clique $\mathcal{C}$.

1.6.1 Exponential Models and Maximum Entropy

The data fitting principle of maximum entropy, which is suitable for learning models from data leads naturally to graphical models in exponential form and also gives nice semantics to the weights in the exponential family. There are many problems with constraints on distributions, but where the information is not complete. For instance, if we knew that for two binary random variables $X, Y \in \{0, 1\}$ and for their paired observations, $2/3$ times X takes value 0, $3/4$ times, Y takes value 1 and if you are asked, to specify the fraction of times $(X, Y) = (0, 0)$, what would you answer with the insufficient specification? Or going back to our diseased network model, if you are given observations on healths of similarly networked people and were to translate these observations into constraints on the moments (and/or the joint quantities), how would you determine the parameters of the probability distribution? There could be many distributions/parameters that are consistent with the observations. What principle should be adopted for making a good choice of the parameters/distribution?

More concretely, say you are given some (feature) functions $f_\alpha(\mathbf{x})$ with $\alpha \in I$ and are also given some observations $\widehat{\mu}_\alpha$ on the expected values of the functions, called *empirical moments*. The observations could come either from data or from physical constraints. We are interested in the family of distributions that are consistent with these constraints.

$$\mathcal{P}(\widehat{\mu}) = \left\{p(.) \left| \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) = \widehat{\mu_\alpha} \quad \forall \alpha \in I \right.\right\}$$

Note that the family could be an empty set if the set of constraints are inconsistent. However, we will assume that the constraints are consistent. We are interested in choosing a particular $\widehat{p}$ from $\mathcal{P}(\mu)$ in a principled manner. Maximum entropy is one such principled method. Entropy is, roughly speaking, a measure of uncertainity or randomness. It has played a vital role in physics, chemisty, statistics, computer science, communication[31], *etc.*

[31]Entropy plays a fundamental role in deciding how far you could compress a sequence of bits.

Definition 15 *The entropy $H(p)$ of the distribution p on a random variable (vector) $\mathbf{X}$ is given by the expected value of the log of the distribution. Depending on whether the random variable (vector) is continuous or discrete, we will have two different definitions of expectation (and therefore for the entropy). For discrete random variable (vector) $\mathbf{X}$,*

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

whereas, for continuous valued random variable $\mathbf{X}$,

$$H(p) = -\int_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x}) d\mathbf{x}$$

It can be easily proved that $H(p) \geq 0$ (convention being that $0 \log 0 = 0$). $H(p) = 0$ if and only if X is deterministically always some a, making $p(a) = 1$ and $p(x) = 0$, $\forall x \neq a$. $H(p)$ is maximal for the uniform distribution.

The principle of maximum entropy can be now defined:

Definition 16 *Given* $\mathcal{P}(\widehat{\mu}) = \left\{ p(.) \middle| \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) = \widehat{\mu_\alpha} \;\; \forall \alpha \in I \right\}$*, choose*

$$\widehat{p} = \underset{p \in \mathcal{P}(\widehat{\mu})}{\text{argmax}} H(p)$$

The intuition here is to balance across the constraints. It is also called the *principle of least commitment* since the goal is to simultaneously respect the data and not commit to anything more.

Theorem 13 *The maximum entropy solution exists and is unique. The unique solution takes the form*

$$p(\mathbf{x}) \propto \exp\left\{ \sum_{\alpha \in I} \eta_\alpha f_\alpha(\mathbf{x}) \right\} \tag{1.49}$$

Proof: The first thing to note is that $\mathcal{P}(\widehat{\mu})$ is a convex[32] (linear), closed[33] and bounded set of distributions. Further, $H(p)$ is continuous, and this, in conjunction with the nature of $\mathcal{P}(\widehat{\mu})$, guarantees that a maximum will be attained. Also, $H(p)$ is strictly concave[34], implying that the maximum will be unique.

The canonical parameters are actually the Langrange multipliers. Consider the Lagrangian $L(p, \eta, \lambda)$:

$$L(p, \eta, \lambda) = H(p) + \sum_{\alpha \in I} \eta_\alpha \left[\widehat{\mu_\alpha} - \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) \right] + \lambda [1 - \sum_{\mathbf{x}} p(\mathbf{x})]$$

[32]What if the set is empty?

[33]Which means the optimum cannot escape to the boundary.

[34]Since $\frac{\partial^2 H}{\partial p^2} = -\frac{1}{p} < 0$.

The KKT necessary and sufficient conditons (see (3.88) on page 242) for optimality of $(\widehat{p(\mathbf{x})}, \widehat{\eta}, \widehat{\lambda})$ yield:

1. $\frac{\partial L}{\partial p} = 0 \Rightarrow \log \widehat{p(\mathbf{x})} - \sum\limits_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x}) - \widehat{\lambda} = 0$. That is,

$$\widehat{p(\mathbf{x})} = \exp\left\{\sum_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x})\right\} e^{\widehat{\lambda}} \tag{1.50}$$

2. $\sum\limits_{\mathbf{x}} \widehat{p(\mathbf{x})} = 1$. Substituting for $p(\mathbf{x})$ from (1.50), we get

$$e^{\widehat{\lambda}} = \frac{1}{\sum\limits_{\mathbf{x}} \exp\left\{\sum\limits_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x})\right\}}$$

   which is a constant.

This proves that $p(\mathbf{x}) \propto \exp\left\{\sum\limits_{\alpha \in I} \eta_\alpha f_\alpha(\mathbf{x})\right\}$. $\square$

This result shows how exponential families can arise in a fairly natural way. It can be further shown that a slight generalization of the maximum entropy principle, called the *Kullkack-Leibler divergence* very naturally leads to the maximum likelihood principle.

Definition 17 *The Kullkack-Leibler (KL) divergence $D(p||q)$ between two distributions $p(.)$ and $q(.)$ is given by the expectation over $p(.)$ of the log-likelihood ratio of $p(.)$ over $q(.)$.*

$$D(p||q) = \sum_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})}$$

For distributions $p(.)$ and $q(.)$ over continuous valued random variables

$$D(p||q) = \int_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x}$$

Like any distance, the KL divergence is always non-negative. Also, $p \equiv q$ if and only if $D(p||q) = 0$. However, by inspection, we can infer that $D(p||q)$ is assymetric, unlike metrics. That is $D(p||q) \neq D(q||p)$. If $q(.)$ were the uniform distribution $D(p||q) = H(p) + c$, where c is some constant.

We will next define a slight generalization of the maximum entropy principle. Recall the definition of $\mathcal{P}(\widehat{\mu})$. Now let q be some additional prior or reference distribution. The generalized definition goes as:

Definition 18 *Given* $\mathcal{P}(\widehat{\mu}) = \left\{ p(.) \,\middle|\, \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) = \widehat{\mu_\alpha} \;\; \forall \alpha \in I \right\}$ *and a reference distribution* $q(\mathbf{x})$*, choose*

$$\widehat{p} = \underset{p \in \mathcal{P}(\widehat{\mu})}{\text{argmin}} D(p||q)$$

The interpretation here is: get as close as possible to the reference distribution, while respecting the data in the form of the constraint set $\mathcal{P}(\widehat{\mu})$. For the maximum entropy principle, the reference distribution happened to be the uniform distribution. Note that we have an 'argmin' here instead of an 'argmax'.

Theorem 14 *The solution to the minimum KL divergence problem in definition 18 exists and is unique. The unique solution takes the form*

$$p(\mathbf{x}) \propto q(\mathbf{x}) \exp\left\{ \sum_{\alpha \in I} \eta_\alpha f_\alpha(\mathbf{x}) \right\}$$

The reference distribution $q(\mathbf{x})$ *plays the role of the function* $h(\mathbf{x})$ *in the exponential form.*

Proof: The proof of existence and uniqueness here are the same as that for the counterpart in the proof of theorem 13. The only change will be the KKT necessary and sufficient conditions for optimality of $(\widehat{p(\mathbf{x})}, \widehat{\eta}, \widehat{\lambda})$ (see (3.88) on page 242).

Consider the Lagrangian $L(p, \eta, \lambda)$:

$$L(p, \eta, \lambda) = D(p||q) + \sum_{\alpha \in I} \eta_\alpha \left[\widehat{\mu_\alpha} - \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) \right] + \lambda[1 - \sum_{\mathbf{x}} p(\mathbf{x})]$$

The KKT necessary and sufficient conditons yield:

1. $\frac{\partial L}{\partial p} = 0 \Rightarrow \log \widehat{p(\mathbf{x})} - \sum_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x}) - \widehat{\lambda} - \log q(\mathbf{x}) = 0$. That is,

$$\widehat{p(\mathbf{x})} = q(\mathbf{x}) \exp\left\{ \sum_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x}) \right\} e^{\widehat{\lambda}} \tag{1.51}$$

2. $\sum_{\mathbf{x}} \widehat{p(\mathbf{x})} = 1$. Substituting for $p(\mathbf{x})$ from (1.50), we get

$$e^{\widehat{\lambda}} = \frac{1}{\sum_{\mathbf{x}} q(\mathbf{x}) \exp\left\{ \sum_{\alpha \in I} \widehat{\eta_\alpha} f_\alpha(\mathbf{x}) \right\}}$$

which is a constant.

This proves that $p(\mathbf{x}) \propto q(\mathbf{x}) \exp\left\{ \sum_{\alpha \in I} \eta_\alpha f_\alpha(\mathbf{x}) \right\}$. □

1.7 Model fitting

Thus far we have discussed graphical models and inference in these models. But how do learn the potential functions associated with a model or the canonical parameters associated with the exponential form, given some data from the real world that describes the phenomena? This opens up a new interesting fundamental question: 'how to infer models from data?'. We will also need a measure of how 'good' a model is and this is often driven by the end goal. These discussions forms the topic of discussion for this section.

Consider a coin tossing experiment, in which $X \sim Ber(\mu)$ where $\mu = \Pr(X = 1)$. Let us say we observe a sequence of coin tosses: $[x^{(1)}, x^{(2)}, \ldots, x^{(m)}]$. Can we use this data to infer something about μ? There are two primary ways to do this.

1. The Bayesian school of thought views the Bayes rule as the fundamental method for inventing model from the data:

$$p(\theta \mid \overline{\mathbf{x}}) = \frac{p(\overline{\mathbf{x}} \mid \theta)p(\theta)}{p(\overline{\mathbf{x}})}$$

   where θ is the underlying model parameter, $p(\overline{\mathbf{x}} \mid \theta)$ is the likelihood term, $p(\theta \mid \overline{\mathbf{x}})$ is the posterior that summarizes the remaining uncertainty in θ, given that you have observed the data $\overline{\mathbf{x}}$. This simple statement has consequences on the way you might view parameters; by imposing a distribution over θ, you are encoding the belief that the parameter itself is a random quantity. This requires viewing all parameters as random variables. The Bayesian technique views μ as a random quantity following a $Beta(a, b)$ distribution[35] on $[0, 1]$.

$$\mu \sim Beta(a, b) \propto \mu^{a-1}(1-\mu)^{b-1}$$

   The Bayesian tries to model the inherent uncertainity in the statistician about the value of the parameter (μ). The Bayesian thus has a more or less fixed procedure for folding in the data into the model.

2. The frequentist's way of measuring probabilities is as numbers measured as outcomes over repeated trials as against the subjective notion of probability adopted by the Bayesians. The frequentist would object to the Bayesian view on several counts: (i) the subjectivity of the procedure; is there a sound justification for the choice of $\mu \sim Beta(a, b)$ and for the particular choice of a and b? (ii) that different results for the same data set could be obtained by different statisticians adhering to different choices of the prior. The frequentist belief is that one should be completely objective with the data; the data should speak so that you get the same model, no matter who builds the model. It does not make sense for the probability

[35]The shape of the Beta distribution depends on the value of the parameters - it could be either uniform or bell-shaped.

of a coin throwing up heads to be a random variable. However, the frequentist does not have any fixed procedure for inventing a model from the data. The frequentist thus considers several estimators for the parameters. One estimator is the sample mean of the data. Any estimator is assessed for its properties such as bias, variability, *etc.* Variability asks: How much would the estimated value vary if the data sample changed? How will the estimate behave if we repeated the experiement several times?

Both Bayesian and frequentist views are compatible with graphical models. Also, the two views agree that there could be different models for different (coin-tossing) experiments. The Bayesian view often gives you a good way of generating good procedures (that is, procedures with good properties) whereas the frequentist view often gives you a very good way of evaluating a procedure.

1.7.1 Sufficient Statistics

Let $\mathbf{X}$ be a random variable. In the present discussion, we will often assume it to correspond to a vector of observations (that is data). Any function $\tau(\mathbf{X})$ is called a statistic. From the Bayesian point of view, a statistic $\tau(\mathbf{X})$ is *sufficient* for the parameter variable θ if $\theta \perp \mathbf{X} \mid \tau(\mathbf{X})$. This relation can be expressed graphically as a markov chain with $\mathcal{V} = \{\mathbf{X}, \tau(\mathbf{X}), \theta\}$, $\mathcal{E} = \{(\mathbf{X}, \tau(\mathbf{X})), (\tau(\mathbf{X}), \theta)$. Intuitively, this means that the function of the observations $\tau(\mathbf{X})$ is what is essential in data to explain the characteristics of θ and that all the dependence between $\mathbf{X}$ and θ is being mediated by $\tau(\mathbf{X})$. $\tau(\mathbf{X})$ is typically much smaller than $\mathbf{X}$ itself.

From the frequentist point of view, θ is an unknown fixed parametrization that we would like to estimate. The sufficiency for a frequentist is that

$$p(\mathbf{x} \mid \tau(\mathbf{x}); \theta) = p(\mathbf{x} \mid \tau(\mathbf{x}))$$

That is, the family of distributions specified by the parametrization θ becomes independent of θ when conditioned on the sufficient statistic $\tau(\mathbf{x})$; it indicates that we have conditioned on sufficient information to capture any information from θ.

A view of sufficiency unified across the Bayesian and frequetist perspectives can be obtained by treating it as a statement about factorization:

$$p(\mathbf{x}, \tau(\mathbf{x}), \theta) = \phi_1(\tau(\mathbf{x}), \theta)\phi_2(\tau(\mathbf{x}), \mathbf{x})$$

It can be proved that this is indeed a unified view of sufficiency, consistent with the Bayesian and frequentist views. Though there is a difference in interpretations, it amounts to the same factorization. Note the similarity of the factorization here with that for general graphical models. As we will see, for the graphical model family, sufficiency properties can be read out, just on the basis of their factorization. Further, the sufficiency property stands our clearly for exponential family. Given iid data[36] $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)})$ each from some

[36]This assumption is common-place because it makes computations very simple and decomposable. It is often not the case, since there could be dependence in the sampling.

exponential family, and given an η (which corresponds to the θ being discussed thus far in the context of model estimation), we can easily infer that

$$p(\overline{\mathbf{x}}_{1,m};\eta) = \prod_{i=1}^{m} p(\mathbf{x}^{(i)};\eta) = \underbrace{\left(\prod_{i=1}^{m} h(\mathbf{x}^{(i)})\right)}_{\phi_2(\tau(\mathbf{x}),\mathbf{x})} \underbrace{\exp\left\{\eta^T\left(\sum_{i=1}^{m}\mathbf{f}(\mathbf{x}^{(i)})\right) - mA(\eta)\right\}}_{\phi_1(\tau(\mathbf{x}),\eta)}$$

This algebra tells us that the quantity $\tau(\overline{\mathbf{x}}) = \sum_{i=1}^{m}\mathbf{f}(\mathbf{x}^{(i)})$ is the sufficient statistic. Note that ϕ_1 can potentially depened on $\tau(\mathbf{x})$, though it does not depend in this case. The key to estimating η are the sufficient statistics $\tau(\overline{\mathbf{x}})$. We will also realize that the prefactor $h(\overline{\mathbf{x}})$ does not play a significant role here.

The quantity

$$\overline{\mu} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{f}(\mathbf{x}^{(i)}) = \sum_{\mathbf{x}} \wp(\mathbf{x})\mathbf{f}(\mathbf{x})$$

is called the *empirical moment parameter* vector, where the empirical distribution $\wp(\mathbf{x})$ is obtained by placing point mass at data points.

$$\wp(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{m}\delta(\mathbf{x}-\mathbf{x}^{(i)})$$

The empirical moment parameters are a special type of empirical moments discussed on page 55. Note that the empirical moment parameters are simply the sufficient statistics scaled down by the number of data points. The empirical moment parameter vector can be contrasted against the moment parameter vector $\mu = [E[f_1(\mathbf{X})], E[f_2(\mathbf{X})], \ldots, E[f_k(\mathbf{X})]]$ for exponential families, as defined on page 49. The two differ in the probability distributions with respect to which they compute their expectations; while the former computes with respect to the empirical distribution, the latter computes with respect to the true distribution.

As an example, consider the Ising model, defined on a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, which is a member of the exponential family. It is distribution is very similar to that of the disease network model (1.47), only difference is that only the first term $x_s x_t$ is retained in the definition of $f_{st}(\mathbf{x})$.

$$p(\mathbf{x},\eta) \propto \exp\left\{\sum_{s\in\mathcal{V}}\eta_s x_s + \sum_{(s,t)\in\mathcal{E}}\eta_{st}x_s x_t\right\}$$

By appropriately identifying the feature function vector $\mathbf{f}$ and η of size $|\mathcal{V}|+|\mathcal{E}|$ each, we can determine the empirical moments to be

$$\widehat{\mu}_s = \frac{1}{m}\sum_{i=1}^{m} x_s^i$$

and

$$\widehat{\mu}_{st} = \frac{1}{m}\sum_{i=1}^{m} x_s^i x_t^i$$

which are just the marginal counts.

What about empirical moments for the reduced parametrization discussed earlier in (1.48) on page 55?

$$p(\mathbf{x}, \eta) = h(\mathbf{x}) \exp\left\{\sum_{\mathcal{C}\in\Pi}\sum_{\alpha\in I_{\mathcal{C}}} \eta_\alpha f_\alpha(\mathbf{x}_{\mathcal{C}}) - A(\eta)\right\}$$

Given iid data $\overline{\mathbf{x}} = \left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right)$, the following factorization holds if each $\mathbf{x}^{(i)}$ follows the distribution specified by the reduced parametrization. Since the apriori information is captured using indicative feature functions, you can have a reduced set of sufficient statistics.

$$p(\overline{\mathbf{x}}_{1,m};\eta) = \prod_{i=1}^{m} p(\mathbf{x}^{(i)};\eta) = \underbrace{\left(\prod_{i=1}^{m} h(\mathbf{x}^{(i)})\right)}_{\phi_2(\tau(\mathbf{x}),\mathbf{x})} \underbrace{\exp\left\{\sum_{\alpha}\eta_\alpha\left(\sum_{i=1}^{m}\sum_{\mathcal{C}\in\Pi|\alpha\in I_{\mathcal{C}}} f_\alpha(\mathbf{x}^{(i)})\right) - mA(\eta)\right\}}_{\phi_1(\tau(\mathbf{x}),\eta)}$$

The form of the empirical moments in this case pools over all cliques that are relevant for a particular feature type f_α (for instance, in a grid, some features may be commonly defined/parametrized across all vertical edges while others across all horizontal edges) and then pools over all data:

$$\widehat{\mu}_\alpha = \sum_{i=1}^{m}\sum_{\mathcal{C}\in\Pi|\alpha\in I_{\mathcal{C}}} f_\alpha(\mathbf{x}^{(i)})$$

The reduced parametrization assumes some kind of homogenity in the model. More precisely, it makes the statistical assumption that the parameter η_α is independent of the exact position in $\mathcal{G}$ and is determined by the local graphical structure. This parametrization assuming homogenity is often desirable since it gets a more stable estimator with less variance even using relatively less data. In fact, the reduced parametrization yields a new exponential family with much lower dimensions, in which the parameters η_α enter in a purely linear way[37]. The issues of when and how to pool data are important ones in modeling.

1.7.2 Maximum Likelihood

The likelihood function interests both Bayesian and frequentists alike. Recall that likelihood $p(\overline{\mathbf{x}} \mid \theta)$ was one part of the Bayes rule. The frequentist interpretation of this quantity is as 'the likelihood of a fixed $\overline{\mathbf{x}}$ as θ varies' whereas the Bayesian interpretation of this quantity is as 'the conditional probability of

[37]A widely used different class of exponential families is called *curved exponential families*, in which the parameters η_α enter in non-linear ways.

a varying $\overline{\mathbf{x}}$ given a fixed θ.' The frequentist thus views likelihood as a function of theta $L(\theta)$ that measures, in some sense, the goodness of θ as it is varied for the particular sample of data in hand. It is often useful to talk about the log-likelihood instead[38].

Definition 19 *Given a sample $\overline{\mathbf{x}} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\}$, the log-likelihood (LL) is defined as*

$$LL(\theta; \overline{\mathbf{x}}) = \frac{1}{m} \log p(\overline{\mathbf{x}} \mid \theta)$$

Note that the $\frac{1}{m}$ term does not really change the optimization and its usage is conventionally just to normalize the log-likelihood.

The maximum likelihood estimate (MLE) is defined next.

Definition 20 *Given a sample $\overline{\mathbf{x}} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\right\}$, the maximum likelihood estimate (MLE) $\widehat{\theta}_{MLE}$ of θ is defined as*

$$\widehat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\ LL(\theta; \overline{\mathbf{x}})$$

As the name suggests, this principle treats log-likelihood as a measure of goodness of the parameter and picks that value which maximizes the LL. Though not against Bayesian principles, MLE has been of greater interest to the frequentists. The estimate $\widehat{\theta}_{MLE}$ is called a point estimate, since the method gives you just a single point.

Let us try to fit (that is, adjust the parameters of) a scalar Gaussian $\mathcal{N}(\mu, \sigma^2)$ to some data $\left\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\right\}$ using the MLE principle. We will assume that the data is iid. This will mean that the joint distribution over the data will factorize into individual data instances, given the parameters.

$$LL(\mu; \overline{\mathbf{x}}) = \frac{1}{m} \sum_{i=1}^{m} \log p(x^{(i)} \mid \mu) = -\frac{1}{2m} \log 2\pi - \frac{1}{2m} \sum_{i=1}^{M} (x^{(i)} - \mu)^2$$

The LL is a quadratic in μ (that is θ), which achieves its maximum at

$$\widehat{\mu}_{MLE} = \frac{1}{m} \sum_{i=1}^{m} x^{(i)}$$

as expected. The data determines the shape of the likelihood and MLE looks for the point $\widehat{\mu}_{MLE}$ that maximizes the LL.

This example of parameter estimation for the simple Gaussian is one of the motivations for using MLE; MLE corresponds to an intuitively appealing estimation for the mean parameter. Secondly MLE has good asymptotic properties.

[38]Since it is a monotonic transformation between the likelihood and log-likelihood, it does not really matter much whether we look at the former or the latter. It is usually more convenient to look at the log-likelihood (LL).

A frequentist takes more interest in the asymptotic properties of the estimator. We can look upon $\widehat{\theta}_{MLE}$ as a random variable, since the data $\overline{\mathbf{x}}$ was itself random and $\widehat{\theta}_{MLE}$ is a function of the data. As the amount of data grows, any reasonable estimator should behave 'well'. One yardstick of good behaviour of the estimator is *consistency*, that is, if there was some true underlying θ^*, we would like

$$\widehat{\theta}_{MLE} \overset{m\rightarrow\infty}{\rightarrow} \theta^*$$

Under most conditions, MLE estimators are consistent. Generally, it can be worrisome if the estimator does not converge to the true answer even with infinite amount of data.

Another yardstick of good behaviour is that asymptotically, the estimator's spread around the true value of the parameter must be Gaussian-like. MLE estimators honor this feature as well; with increasing m, the distribution of $\widehat{\theta}_{MLE}$ tends to be Gaussian with the true mean θ^* and the spread Σ given by the Fisher information matrix[39].

In small sample regimes (that is, if you do not have too much data), MLE does not behave well. In such settings, the frequentist adds a penalty term, called the *regularization* term (such as μ^2 in the above example) to the likelihood objective to somehow prevent the estimator from drifting away based on a small m.

1.7.3 What the Bayesians Do

The Bayesians focus more on the posterior $p(\theta \mid \overline{\mathbf{x}})$, which is the likelihood multiplied by a prior.

$$p(\theta \mid \overline{\mathbf{x}}) \propto p(\overline{\mathbf{x}} \mid \theta)p(\theta)$$

For example, let $x^{(i)} \sim \mathcal{N}(\mu, 1)$ for $1 \leq i \leq m$ and let $\mu \sim \mathcal{N}(\nu, \sigma^2)$. The parameter μ is a random variable, whereas the *hyperparameters* μ' and σ are fixed. The choice of the values μ' and σ could significantly influence the posterior. There is a class of models[40] called the 'hierarchical Bayes models' that put a prior on the hyper-parameters, priors again on the hyper-hyper-parameters and so on. It stops (for pragmatic reasons) when the abstration has been sufficiently performed so that it matters little what the fixed values of the hyper^n-parameters are.

The posterior for this example, takes the form

$$p(\theta \mid \overline{\mathbf{x}}_{1,m}) \propto \exp\left\{-\frac{1}{2}\sum_{i=1}^{m}(x^{(i)} - \mu)^2\right\} \exp\left\{-\frac{1}{2\sigma^2}(\mu - \nu)^2\right\}$$

The Bayesian is not interested just in the point estimate, but moreso in the entire posterior distribution. In this case (a special instance of Kalman filters) the posterior again turns out to be a Gaussian; $\mu_{\overline{\mathbf{x}}_{1,m}} \sim \mathcal{N}\left(E(\mu \mid \overline{\mathbf{x}}_{1,m}), Var(\mu \mid \overline{\mathbf{x}}_{1,m})\right)$.

[39]Related to this is the Cramer-Rao lower bound.

[40]The broader area is called *Robust Bayesian statistics.*

A little calculation should convince you that $\mu_{\overline{\mathbf{x}}_{1,m}} \sim \mathcal{N}\left(\frac{m\sigma^2}{m\sigma^2+1}\widehat{\mu}_{MLE} + \frac{1}{m\sigma^2+1}\nu, Var(\mu \mid \overline{\mathbf{x}}_{1,m})\right)$. How does the posterior behave as $m \to \infty$? We can easily see that with increasing amount of data, the likelihood will completely swamp the prior. That is, $E(\mu \mid \overline{\mathbf{x}}_{1,m}) \overset{m\to\infty}{\to} \widehat{\mu}_{MLE}$. So the choice of the hyper-parameters is irrelevant as the size of the data goes to ∞. Thus, MLE behaves well asymptotically even from the Bayesian point of view. Unlike the frequentist, the Bayesian does not need to explicitly handle the small sample regimes.

In general, the posterior might not have a finite parametrization. In such cases, the Bayesians do compute point estimates. Examples of point estimates computed by Bayesians are:

1. *Bayes estimate:* This is the mean of the posterior:

$$\widehat{\theta}_{Bayes} = \int \theta p(\theta \mid \overline{\mathbf{x}}_{1,m}) d\theta$$

2. *MAP estimate:* This is the mode of the posterior:

$$\widehat{\theta}_{MAP} = \underset{\theta}{\operatorname{argmax}} \frac{1}{m} \log \{p(\theta \mid \overline{\mathbf{x}}_{1,m})\}$$

The MAP estimate is very much related to the MLE. Invoking the Bayes rule and ignoring the term involving $\log p(\overline{\mathbf{x}}_{1,m})$,

$$\begin{aligned} \widehat{\theta}_{MAP} &= \underset{\theta}{\operatorname{argmax}} \frac{1}{m} \log \{p(\overline{\mathbf{x}}_{1,m} \mid \theta)\} + \frac{1}{m} \log \{p(\theta)\} \qquad (1.52) \\ &\underset{\text{iid}}{=} \underset{\theta}{\operatorname{argmax}} \frac{1}{m} \sum_{i=1}^{m} \log \left\{p(x^{(i)} \mid \theta)\right\} + \frac{1}{m} \log \{p(\theta)\} \end{aligned}$$

The first term in the decomposition is the data likelihood term, while the second term is the prior term. Let us study this decomposition.

(a) This decomposition suggests that if the prior is uniform, $\widehat{\theta}_{MAP} = \widehat{\theta}_{MLE}$.

(b) Secondly, in the asymptotic case, as $m \to \infty$, the prior component fades away (since it has no dependence on m in the numerator) in contrast to the likelihood term[41]. In fact, as $m \to \infty$, $\widehat{\theta}_{MAP} \to \widehat{\theta}_{MLE}$.

(c) If the prior is assumed to be Gaussian, then the prior term will assume the form of a quadratic penalty. This is the same as a quadratic regularization term for MLE. With different choices of priors on θ, you get different regularizations. For example, a Laplacian prior on θ results in $L1$ regularization. For example, Lasso is L1 regularization for a regression problem. How the choice of regularization affects these estimators is an area of active research.

[41]One could use the central limit theorem or law of large numbers to study the behaviour of the likelihood term as $m \to \infty$.

1.7.4 Model Fitting for Exponential Family

We will initiate the discussion of model fitting in exponential families by looking at maximum likelihood (ML) estimation. Just as for most other properties of exponential models, there is something crisper that we can state about the ML properties of exponential models. The optimization conditions will turn out to take special and easily interpretable forms.

Recall the specification of any member of the exponential family from (1.43).

$$p(\mathbf{x} \mid \eta) = h(\mathbf{x}) \exp\left\{\eta^T \mathbf{f}(\mathbf{x}) - A(\eta)\right\}$$

We will now see how the empirical moment parameters, defined on page 61 and discussed subsequently become relevant for parameter estimation. Given iid observations $\overline{\mathbf{x}} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\right\}$, the normalized LL for this distribution decouples as a sum

$$\begin{aligned}
LL(\eta; \overline{\mathbf{x}}) &= \frac{1}{m} \log p(\overline{\mathbf{x}} \mid \eta) && (1.53)\\
&= \frac{1}{m}\left\{\eta^T \sum_{i=1}^{m} \mathbf{f}(\mathbf{x}^{(i)}) - mA(\eta)\right\} + \log h(\overline{\mathbf{x}}) && (1.54)\\
&= \left\{\frac{1}{m}\eta^T \sum_{i=1}^{m} \mathbf{f}(\mathbf{x}^{(i)})\right\} - A(\eta) + \log h(\overline{\mathbf{x}}) && (1.55)\\
&= \left\{\eta^T \overline{\mu}(\overline{\mathbf{x}})\right\} - A(\eta) + \log h(\overline{\mathbf{x}}) && (1.56)
\end{aligned}$$

where $\overline{\mu}(\overline{\mathbf{x}}) = \frac{1}{m}\sum_{i=1}^{m} \mathbf{f}(\mathbf{x}^{(i)})$ is the vector of empirical moment parameters. Since $h(\overline{\mathbf{x}})$ is not a function of the parameter (and instead, is a constant offset), it follows that the MLE for the exponential family takes the form

$$\widehat{\mu}_{ML} \in \underset{\eta \in Domain(\eta)}{\text{argmax}} \left\{\eta^T \overline{\mu}(\overline{\mathbf{x}}) - A(\eta)\right\}$$

This is a very crisp formulation of sufficiency; given data, all you need to compute are $\overline{\mu}(\overline{\mathbf{x}})$, forget the data $\overline{\mathbf{x}}$ and focus on solving the optimization problem. Since $A(\eta)$ is infinitely differentiable and since the remainder of the objective is linear, we could make use of the first order necessary optimality conditions from (3.44) on page 216:

$$\nabla LL(\eta; \overline{\mathbf{x}}) = 0$$

That is,

$$\overline{\mu}(\overline{\mathbf{x}}) - \nabla A(\eta) = \overline{\mu}(\overline{\mathbf{x}}) - E(\mathbf{f}(\mathbf{X})) = 0$$

where, we may recall from (1.44) on page 49 that $\nabla A(\eta)$ is the vector of moments predicted by the model. This important condition

$$\overline{\mu}(\overline{\mathbf{x}}) = E(\mathbf{f}(\mathbf{X})) \tag{1.57}$$

is called the *moment matching condition*[42] and it states that for maximizing the likelihood, we need to match the empirical moments to the model moments. In general, these coupled d equations are complex and non-linear ones, that are not very easy to solve, though there are some exceptions. The necessary conditions suggest that the parameters should be learnt in such a way that if you drew samples from the model, they should look like the given data.

As example, let X be a bernoulli random variable. Recollect the specification of the bernoulli distribution as a member of exponential family from page 1.4: $\eta = \log \frac{\mu}{1-\mu}$, $f(x) = x$ and $A(\eta) = \log\{1 + e^{\eta}\}$. This leads to the following moment matching conditions

$$\overline{\mu} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)} = A'(\eta) = \frac{e^{\eta}}{1+e^{\eta}}$$

The empirical moments is the fraction of heads that come up in the m coin tosses of the experiment. The expression for η will be

$$\eta = \log\left(\frac{\overline{\mu}}{1-\overline{\mu}}\right)$$

which corresponds to the logs-odd ratio. When does this have a solution? It has a solution if $\overline{\mu} \in (0, 1)$ but not if $i = 1$ or $i = 0$, which can happen if all the coin tosses landed up with heads or tails[43]. If $\overline{\mu} \in (0, \frac{1}{2})$, $\eta < 0$ and if $\overline{\mu} \in (\frac{1}{2}, 1)$, $\eta > 0$. So if $\overline{\mu} = 1$, strictly speaking the MLE does not exist. Note that if we were to use the original formulation of the bernoulli distribution on page 48, moment matching would have yielded the MLE as $\mu = \overline{\mu}$. For large amounts of data, using the weak law of large numbers, you can be convinced that this loss can be recovered using even the exponential formulation.

As another example, let us revist the multivariate normal distribution in the canonical form first uncovered in (1.46):

$$p(\mathbf{x}, \eta, \Omega) = \exp\left\{\eta^T\mathbf{x} - \frac{1}{2}trace(\Omega\mathbf{x}\mathbf{x}^T) + \mathbf{x}\right\}$$

where

$$\eta = \Sigma^{-1}\mu$$

and

$$\Omega = \Sigma^{-1}$$

[42]This is related to moment matching rooted in classical statistics. Though not related directly to maximum likelihood, these two boil down to the same criteria for the exponential family.

[43]This is not impossible, especially if the tossing is done by some magician such as Persi Warren Diaconis or by a machine built by him.

Moment matching will yield the model first order and second order moments in terms of the empirical mean and the empirical second order moment matrix respectively. That is,

$$E[\mathbf{x}] = \mu = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)} = \overline{\mu}$$

and

$$E[\mathbf{x}\mathbf{x}^T] = \Sigma + \mu\mu^T = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)}\left(\mathbf{x}^{(i)}\right)^T$$

Solving this system, we get Ω_{ML} as the inverse of the sample covariance matrix

$$\Omega_{ML} = \left(\frac{1}{m}\sum_{i=1}^{m}\mathbf{x}^{(i)}\left(\mathbf{x}^{(i)}\right)^T - \overline{\mu}\mu^T\right)^{-1}$$

(which exists if the sample covariance matrix has full rank) and

$$\eta_{ML} = \Omega_{ML}\overline{\mu}$$

In practice (such as in text mining problems), it can happen that many features are never observed and consequently, the corresponding empirical moment parameters will be 0. Similarly, for the multivariate Gaussian example, if you do not have sufficient data, the sample covariance matrix may not have full rank. More precisely, we need to pay heed to $Domain(\eta)$, since the optimal values could reside on the boundary. This is addressed using regularization through a penalized log-likelihood objective.

1.7.5 Maximum Likelihood for Graphical Models

For general large graphical models, the moment matching is not at all easy to solve. It has therefore been a practice to often resort to iterative algorithms for solving the set of moment matching equations. We will illustrate this difficulty through the Ising model (*c.f.* page 61), which is a model on an undirected graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$.

$$p(\mathbf{x}, \eta) = \exp\left\{\sum_{s\in\mathcal{V}}\eta_s x_s + \sum_{(s,t)\in\mathcal{E}}\eta_{st}x_s x_t - A(\eta)\right\}$$

Let us say we have iid data $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)})$. The empirical moments are

$$\widehat{\mu}_s = \frac{1}{m}\sum_{i=1}^{m}x_s^i$$

and

$$\widehat{\mu}_{st} = \frac{1}{m}\sum_{i=1}^{m}x_s^i x_t^i$$

The $d = |\mathcal{V}| + |\mathcal{E}|$ moment matching equations will be

$$\sum_{\mathbf{x}\in\{0,1\}^{|\mathcal{V}|}} \exp\left\{\sum_{s\in\mathcal{V}} \eta_s x_s + \sum_{(s,t)\in\mathcal{E}} \eta_{st} x_s x_t - A(\eta)\right\} x_s = \widehat{\mu}_s$$

and

$$\sum_{\mathbf{x}\in\{0,1\}^{|\mathcal{V}|}} \exp\left\{\sum_{s\in\mathcal{V}} \eta_s x_s + \sum_{(s,t)\in\mathcal{E}} \eta_{st} x_s x_t - A(\eta)\right\} x_s x_t = \widehat{\mu}_{st}$$

The task is to estimate the values of the $|\mathcal{V}| + |\mathcal{E}|$ sized vector η. The log-normalization constant $A(\eta)$ is in turn very complex and involves

$$A(\eta) = \sum_{x\in\{0,1\}^{|\mathcal{V}|}} \exp\left\{\sum_{s\in\mathcal{V}} \eta_s x_s + \sum_{(s,t)\in\mathcal{E}} \eta_{st} x_s x_t - A(\eta)\right\}$$

In general, though the equations are intuitive (asserting that parameters be picked to match the data), solving them is very very complex. What if the graph were a tree? Then the solution can be obtained efficiently, as we will see subsequently. Also, $A(\eta)$ as well as the marginal $p(\mathbf{x};\eta)$ could be computed in polynomial time leading to efficient inference as well. Thus, solving the moment matching equations is closely linked to the graph structure. In general, solving the estimation problem inevitably involves solution to the marginalization problem, which can at least be performed efficiently for tree structured models.

Let us consider some graphical models, whose estimation problems are easy. Consider a markov chain with $\mathcal{V} = \{X_1, X_2, X_3, X_4\}$, $\mathcal{E} = \{(X_1, X_2), (X_2, X_3), (X_3, X_4)\}$. Though it can be thought of as a directed model, we will choose an exponential factorization that will link it to the general exponential machinery. Let each X_i have k possible states. Also, let

$$\lambda_{st} = \sum i = 1, j = 1^k \lambda_{st,ij}(\delta(x_s, i)\delta(x_t, j))$$

where $\delta(x,i)\delta(y,j)$ is an indicator feature function for each fixed (i,j) pair and is expressible as a $k \times k$ matrix. $\lambda_{st,ij}$'s are the canonical parameters. We can then adopt the exponential form

$$p(\mathbf{x};\lambda) = \exp\left\{\lambda_{12}(x_1, x_2) + \lambda_{23}(x_2, x_3) + \lambda_{34}(x_3, x_4)\right\}$$

Given data $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)})$, the empirical moment parameters (sufficient statistics scaled down by m) can be computed to be

$$\overline{\mu}_{st}(x_s, x_t) = \frac{1}{m}\sum_{i=1}^{m} \delta(x_s = x_s^{(i)})\delta(x_t = x_t^{(i)})$$

for $(s,t) \in \{(1,2), (2,3), (3,4)\}$ and

$$\overline{\mu}_s(x_s) = \frac{1}{m}\sum_{i=1}^{m} \delta(x_s = x_s^{(i)})$$

for $s \in \{1, 2, 3, 4\}$.

These simply correspond to the empirical marginal probability $\wp(x_s, x_t)$. The moment matching conditions can be satisfied[44] by the following assignments to the canonical parameters:

$$\widehat{\lambda}_{12,12} = \log \overline{\mu}_{12}(x_1, x_2)$$

$$\widehat{\lambda}_{23,23} = \log \frac{\overline{\mu}_{23}(x_2, x_3)}{\overline{\mu}_2(x_2)}$$

$$\widehat{\lambda}_{34,34} = \log \frac{\overline{\mu}_{34}(x_3, x_4)}{\overline{\mu}_3(x_3)}$$

Thus, the canonical parameters can be computed in closed form. The task is similarly simple for trees - the basic intuition has been captured in this markov chain example. However, the simple solution suggested in this four node example does not hold for a cyclic graph having 4 nodes with edges defined as $\mathcal{E} = \{(X_1, X_2), (X_2, X_3), (X_3, X_4), (X_4, X_1)\}$.

The task is also equally simple, with closed form solutions for the class of decomposable graphs (which could be graphs with cycles), defined on page 31 in definition 12.

1.7.6 Iterative Algorithms

We will now move to more general graphs and describe iterative algorithms for solving fixed point equations. We will assume that the potential functions are defined on maximal cliques.

$$p(\mathbf{x}, \phi) \propto \prod_{\mathcal{C} \in \Pi} \phi_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) \equiv \exp\left\{\sum_{\mathcal{C} \in \Pi} \theta_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}})\right\}$$

The general setting for maximum likelihood optimization is that we have to solve a set of fixed point equations

$$\mathbf{F}(\eta) = E_\eta[\mathbf{f}(\mathbf{X})] - \overline{\mu} = \mathbf{0}$$

(such as $\widehat{\mu}_\alpha = \sum_{i=1}^{m} \sum_{\mathcal{C} \in \Pi | \alpha \in I_{\mathcal{C}}} f_\alpha(\mathbf{x}^{(i)})$ in the case of reduced parametrization - *c.f.* page 62). For this system, we will investigate an iterative solution of the form

$$\eta^{(t+1)} = \eta^{(t)} + f\left(\eta^{(t)}\right)$$

so that at the fixed point $\eta^{(t^*)}$, $f\left(\eta^{(t^*)}\right) = 0$ and therefore $\eta^{(t^*+1)} = \eta^{(t^*)}$. t is the time step.

This general algorithm is called Iterative Proportional Fitting (IPF) and goes as follows for a general undirected graph $\mathcal{G} = <\mathcal{V}, \mathcal{E}>$:

[44]EXERCISE: Prove.

1. Start with some initial factorization (which could be uniform)

$$p(\mathbf{x}) \propto \prod_{\mathcal{C}\in\Pi} \phi_{\mathcal{C}}^{(0)}(\mathbf{x}_{\mathcal{C}}) \equiv \exp\left\{\sum_{\mathcal{C}\in\Pi} \theta_{\mathcal{C}}^{(0)}(\mathbf{x}_{\mathcal{C}})\right\}$$

2. For $t = 1$ onwards, let $\mathcal{C}$' range stepwise over the cliques in Π. Update

$$\phi_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \phi_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})\frac{\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})}{\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})} \tag{1.58}$$

where, $\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})$ is the current model marginal, computed as[45]

$$\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'}) = \frac{1}{Z_{\phi^{(t)}}}\sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} p(\mathbf{x};\phi^{(t)})$$

While the model marginals can be computed efficiently for tree structured or even decomposable graphs using message passing formalisms, we may have to resort to approximate inferencing for general graphs. $\overline{\mu}_{\mathcal{C}'}$ are the empirical marginals that are precomputed from data $\overline{\mathbf{x}}$ as

$$\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'}) = \frac{1}{m}\sum_{i=1}^{m}\delta\left(\mathbf{x}_{\mathcal{C}'},\mathbf{x}_{\mathcal{C}'}^{(i)}\right)$$

Equivalently one may also use the following update rule:

$$\theta_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \theta_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'}) + \log\frac{\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})}{\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})} \tag{1.59}$$

The algorithm is called *iterative proportional fitting* because at every step, the potentials are updated proportional to how well the empirical moment parameters fit the model moments for a clique $\mathcal{C}'$.

Convergence of the iterative algorithm

It is easy to see that at the fix point $t = t^*$, the algorithm will yield the MLE $\phi^{(t^*)}$ since, for all $\mathcal{C}\in\Pi$,

$$\mathbf{F}(\phi^{(t^*)}) = \mu_{\mathcal{C}}^{(t^*)}(\mathbf{x}_{\mathcal{C}}) - \overline{\mu}_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) = 0$$

[45]You could formulate this problem using a set of $\delta_{\mathcal{C}}$ feature functions, and $\lambda_{\mathcal{C},\mathbf{v}_{\mathcal{C}}}$ canonical parameters, as was adopted in the example on page 69. Here, $\mathbf{v}_{\mathcal{C}}$ is a configuration vector that could be assumed by variables on clique $\mathcal{C}$.

But we also need to show that the updates will always converge to the fix point. We first observe that after updating using step (1.59) (or equivalently, (1.58)), the moment matching will be achieved for clique $\mathcal{C}'$.

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})$$

Why is this so? By definition,

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \frac{1}{Z_{\theta^{(t+1)}}} \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} p(\mathbf{x};\theta^{(t+1)}) = \frac{1}{Z_{\theta^{(t+1)}}} \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} \exp\left\{\theta_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'}) + \sum_{\mathcal{C}\in\Pi,\mathcal{C}\neq\mathcal{C}'} \theta_{\mathcal{C}}^{(t)}(\mathbf{x}_{\mathcal{C}})\right\}$$

That is, every factor that is not in $\mathcal{C}'$ is not changing with respect to the previous step. Expanding upon this, using the update equation (1.59),

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \frac{1}{Z_{\theta^{(t+1)}}} \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} \exp\left\{\theta_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'}) + \log\frac{\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})}{\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})} + \sum_{\mathcal{C}\in\Pi,\mathcal{C}\neq\mathcal{C}'} \theta_{\mathcal{C}}^{(t)}(\mathbf{x}_{\mathcal{C}})\right\} = \frac{1}{Z_{\theta^{(t+1)}}} \frac{\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})}{\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})} \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} \mathrm{ex}$$

since $\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})$ and $\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})$ are independent of the inner summation. This leads to

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \frac{Z_{\theta^{(t)}}}{Z_{\theta^{(t+1)}}} \frac{\overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})}{\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'})} \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} p(\mathbf{x}_{\mathcal{C}'};\theta^{(t)}$$

By definition,

$$\mu_{\mathcal{C}'}^{(t)}(\mathbf{x}_{\mathcal{C}'}) = \sum_{\mathbf{x}_{\mathcal{V}\setminus\mathcal{C}'}} p(\mathbf{x}_{\mathcal{C}'};\theta^{(t)}$$

Thus,

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \frac{Z_{\theta^{(t)}}}{Z_{\theta^{(t+1)}}} \overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})$$

Since both empirical and model moments are expected to sum to 1 across all cliques in the graph, we should have

$$1 = \sum_{\mathcal{C}'\in\Pi} \mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \sum_{\mathcal{C}'\in\Pi} \frac{Z_{\theta^{(t)}}}{Z_{\theta^{(t+1)}}} \overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'}) = \frac{Z_{\theta^{(t)}}}{Z_{\theta^{(t+1)}}}$$

Consequently, we will have that once the update step (1.59) is applied, moment matching will hold on $\mathcal{C}'$.

$$\mu_{\mathcal{C}'}^{(t+1)}(\mathbf{x}_{\mathcal{C}'}) = \overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})$$

This takes us one step forward. But are these step updates guranteed to lead to convergence? To see that convergence holds, we will note that the IPF algorithm is an instance of the coordinate-wise ascent algorithm (*c.f.* page 247). This becomes obvious by stating the log-likelihood objective

$$LL(\theta;\overline{\mathbf{x}}) = \frac{1}{m}\left(\sum_{\mathcal{C}\in\Pi} \theta_{\mathcal{C}} \overline{\mu}_{\mathcal{C}'}(\mathbf{x}_{\mathcal{C}'})\right) - A(\{\theta_{\mathcal{C}} \mid \mathcal{C}\in\Pi\})$$

and viewing it as a function of $\theta_{\mathcal{C}'}$

$$g(\theta_{\mathcal{C}'}) = LL\left(\theta_{\mathcal{C}'}; \left\{\theta_{\mathcal{C}} = \theta_{\mathcal{C}}^{(t)} \mid \mathcal{C} \neq \mathcal{C}'\right\}, \overline{\mathbf{x}}\right)$$

The first order necessary optimality condition for $g(\theta_{\mathcal{C}'})$ precisely yield the moment matching equation for $\mathcal{C}'$. Since $g(\theta_{\mathcal{C}'})$ can be shown to be concave, the first order necessary conditions are also sufficient. The fact that the application of coordinate descent in this setting will lead to convergence follows from the fact that $LL(\theta; \overline{\mathbf{x}})$ is strongly concave.

1.7.7 Maximum Entropy Revisted

The IPF algorithm exploited the idea of matching moments in order to maximize likelihood. Recall from Section 1.6.1, the maximum entropy principle that seeks to find $\wp(.) \in \mathcal{P}(\widehat{\mu})$ that maximizes $H(p)$. $\mathcal{P}$ is the set of all distributions that satisfy empirical moment constraints $\widehat{\mu_\alpha}$:

$$\mathcal{P}(\widehat{\mu}) = \left\{ p(.) \left| \sum_{\mathbf{x}} p(\mathbf{x}) f_\alpha(\mathbf{x}) = \widehat{\mu_\alpha} \quad \forall \alpha \in I \right.\right\}$$

The problem of maximum entropy was also shown to be equivalent to the problem of minimizing the KL divergence between p and the reference uniform distribution u:

$$\widehat{p}_{ME} = \underset{p \in \mathcal{P}(\widehat{\mu})}{\text{argmax}}\ H(p) = \underset{p \in \mathcal{P}(\widehat{\mu})}{\text{argmin}}\ D(p||u) \tag{1.60}$$

On the other hand, the problem of maximum likelihood estimation problem is specified as

$$\widehat{p}_{MLE} = \underset{\eta}{\text{argmax}}\ \frac{1}{m} \sum_{i=1}^{m} \log p(\mathbf{x}^{(i)}; \eta)$$

To help us establish the connection between the two, we will define the data $\overline{\mathbf{x}}$ driven empirical distribution as

$$\wp_{\overline{\mathbf{x}}}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} \delta(\mathbf{x} = \mathbf{x}^{(i)})$$

This definition yields an alternative expression for the MLE as the minimizer of the KL divergence between the empirical distribution and the model distribution.

$$\begin{aligned}
\widehat{p}_{MLE} &= \underset{\eta}{\text{argmax}}\left(\frac{1}{m}\sum_{i=1}^{m}\wp_{\overline{\mathbf{x}}}\log p(\mathbf{x}^{(i)};\eta)\right) && (1.61)\\
&= \underset{\eta}{\text{argmax}}\left(\frac{1}{m}\sum_{i=1}^{m}\wp_{\overline{\mathbf{x}}}\log p(\mathbf{x}^{(i)};\eta)\right)+\left(\frac{1}{m}\sum_{i=1}^{m}\wp_{\overline{\mathbf{x}}}\log \wp_{\overline{\mathbf{x}}}\right) && (1.62)\\
&= \underset{\eta}{\text{argmax}} - D\left(\wp_{\overline{\mathbf{x}}}||p(\overline{\mathbf{x}};\eta)\right)\\
&= \underset{\eta}{\text{argmin}} D\left(\wp_{\overline{\mathbf{x}}}||p(\overline{\mathbf{x}};\eta)\right)\\
&= \underset{p\in\mathbf{E}(\mathbf{f})}{\text{argmin}} D\left(\wp_{\overline{\mathbf{x}}}||p(.)\right) && (1.63)
\end{aligned}$$

where, $\mathbf{E}(\mathbf{f})$ is the exponential family of distributions having $\mathbf{f}$ as the vector of feature functions that also figure in the specification of constraints in $\mathcal{P}\left(\overline{\mu}\right)$:

$$\mathbf{E}(\mathbf{f}) = \left\{p(.)\left|p(\mathbf{x};\eta) \propto \exp\left\{\eta^T\mathbf{f}(\mathbf{x})\right\}\right.\right\}$$

We will now show the equivalence of specifications in (1.60) and (1.63). The discussion so far will be relevant in our proof. On the one hand, we have seen in (1.49) in theorem 13 that the constraint in $\mathbf{E}(\mathbf{f})$ is satisified by any solution to (1.60). While on the other hand, we know that the moment matching conditions for (1.63) in (1.57) are precisely the constraints in $\mathcal{P}(\overline{\mu})$.

Theorem 15 $\widehat{p}_{MLE} = \widehat{p}_{ML}$ *for exponential families, where:*

$$\widehat{p}_{MLE} = \underset{p\in\mathbf{E}(\mathbf{f})}{\text{argmin}} D\left(\wp_{\overline{\mathbf{x}}}||p(.)\right)$$

and

$$\widehat{p}_{ME} = \underset{p\in\mathcal{P}(\widehat{\mu})}{\text{argmin}}\ D(p||u)$$

Two differences between these formulations are:

1. *While $\widehat{p}_{MLE}$ involves optimization over the second argument in the KL divergence, $\widehat{p}_{ME}$ involves optimization over the first argument.*

2. *The entry point of the data is also toggled in the two; while $\widehat{p}_{ME}$ has data entering through constraint set $\mathcal{P}$, $\widehat{p}_{MLE}$ has data entering through the cost function.*

This is characteristic of dual problems.

Proof: This can be proved by first showing that the problem in (1.60) is the dual of (1.63). Next we will need to apply duality theory in Theorem 67, which states that for convex cost function and convex inequality constraint set, the KKT conditions are necessary and sufficient conditions for zero duality gap. It can be proved that (1.60) is convex and that the parameters for $\widehat{p}_{MLE}$ are solutions to the KKT conditions.

The theorem can also be proved by invoking the so called *pythagorean theorem*[46] for general class of distance that includes distributions. In this particular case, it can be shown that for all $\widehat{p} \in \mathcal{P}(\overline{\mu}) \cap \mathbf{E}(\mathbf{f})$ and for all $p_{\mathcal{P}} \in \mathcal{P}(\overline{\mu})$ and $p_{\mathcal{E}} \in \mathcal{E}(\mathbf{f})$,

$$D(p_{\mathcal{P}}||p_{\mathbf{E}}) = D(p_{\mathcal{P}}||\widehat{p}) + D(\widehat{p}||p_{\mathbf{E}})$$

If $\widehat{q}, \widehat{p} \in \mathcal{P}(\overline{\mu}) \cap \mathbf{E}(\mathbf{f})$, then it will turn out by simple application of the theorem that $D(\widehat{q}||\widehat{p}) + D(\widehat{p}||\widehat{q}) = 0$, which implies that $\widehat{p} = \widehat{q}$. That is, $\mathcal{P}(\overline{\mu}) \cap \mathbf{E}(\mathbf{f})$ is a singleton and $\widehat{p}$ should correspond to both $\widehat{p}_{MLE}$ and $\widehat{p}_{ML}$. $\square$

1.8 Learning with Incomplete Observations

Thus far, we have focussed on learning parameters for graphical models using complete observations; the underlying model was $p(\mathbf{x}; \theta)$ and the observations (data) were $\overline{\mathbf{x}} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)})$. An example of such a learning task was presented in the case of Markov chains on page 69. Consider the hidden markov model from page 46 with $\mathcal{V} = \{X_1, X_2, X_3, X_4, X_5, Y_1, Y_2, Y_3, Y_4, Y_5\}$ and $\mathcal{E} = \{(X_1, X_2), (X_1, Y_1), (X_2, X_3), (X_2, Y_2), (X_3, X_4), (X_3, Y_3), (X_4, X_5), (X_4, Y_4), (X_5, Y_5)\}$. where nodes $\{X_1, X_2, X_3, X_4, X_5\}$ are hidden variables and nodes $\{Y_1, Y_2, Y_3, Y_4, Y_5\}$ are the observed variables.

Mixture models are another set of popular example, which feature hidden variables. Many real world problems are characterized by distinct classes/subpopulations that the data falls into and can be modeled using mixture models.

Definition 21 *Let $Z \in \{z_1, z_2, \ldots, z_k\}$ be a multinomial variable indicating mixture component. Let $\mathbf{X}$ be a random variable (vector), whose distribution is specified, conditioned on different values z_i of Z as*

$$p(\mathbf{x}|z_i; \theta_i) \sim f_i(x; \theta_i)$$

Then the finite mixture model is defined as

$$p(\mathbf{x}) \sum_{i=1}^{k} p(z_i) f_i(\mathbf{x}; \theta_i)$$

with k being the number of mixture components, Z called the mixture indicator component, $f_i(\mathbf{x}; \theta_i)$ termed as the density of the i^{th} mixture component with parameters θ_i. The quantities $p(z_i) = \pi_i$ are also called mixing weights, representing the proportion of the population in subpopulation i. Thus, $\pi = [\pi_1, \pi_2, \ldots, \pi_k]$ and $\theta = [\theta_1, \theta_2, \ldots, \theta_k]$ are the paramaters of a finite mixture model, with a fixed value of k. As a graphical model, the mixture model can be represented as a two node graph: $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ with $\mathcal{V} = \{\mathbf{X}, Z\}$ and $\mathcal{E} = \{(\mathbf{X}, Z)\}$.

[46]The right angle here is not the conventional one, but a notional one.

As an example, the density of each mixture component could be Gaussian with $\theta_i = (\mu_i, \Sigma_i)$.

$$f_i(\mathbf{x};\theta_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$$

The distribution $p(\mathbf{x})$ is then called a mixture of Gaussians. In general, it not Gaussian itself.

How can the parameters be learnt in the presence of incomplete data? In the case of the HMM example, we might be provided only with observations $\overline{\mathbf{y}}$ for $\mathbf{Y}$ and expected to learn the parameters. Or in the case of mixture models, we might be presented only with instances of $\mathbf{X}$ in the form of $\overline{\mathbf{x}}$ and required to learn parameters π and θ. We will illustrate parameter learning for the mixture model problem.

1.8.1 Parameter Estimation for Mixture Models

It can be shown that learning for mixture models is an easy problem if the data is fully observed in the form $(\overline{\mathbf{x}}, \overline{z}) = \left[(\mathbf{x}^{(1)}, z^{(1)}), (\mathbf{x}^{(2)}, z^{(2)}), \ldots, (\mathbf{x}^{(m)}, z^{(m)})\right]$. The joint distribution can be decomposed as

$$p(\mathbf{x}, z; \theta) = p(z)p(\mathbf{x} \mid z, \theta)$$

If $p(\mathbf{x} \mid z, \theta)$ is Gaussian and since $p(z)$ is multinomial, the joint will be in exponential form with Gaussian and multinomial sufficient statistics. The maximum likelihood estimation will boil down to moment matching with respect to these sufficient statistics, leading to an easy estimation problem.

In the incomplete data setting, we are given only $\overline{\mathbf{x}}$ while observations $\overline{z}$ on the mixture components are hidden. The likelihood can still be expressed and maximized:

$$LL(\pi, \theta; \overline{\mathbf{x}}) = \frac{1}{m}\sum_{i=1}^{m} \log p(\mathbf{x}^{(i)};\theta) = \frac{1}{m}\sum_{i=1}^{m} \log \left[\sum_{j=1}^{k} \pi_j f_j(\mathbf{x}^{(i)};\theta_j)\right]$$

subject to the constraints that $\pi_j \geq 0$ and $\sum_{j=1}^{k} \pi_j = 1$.

Unfortunately, log cannot be distributed over a summation and that creates the main bottleneck. In case the densities are Gaussians, the objective to be maximized will be

$$LL(\pi, \mu, \Sigma; \overline{\mathbf{x}}) = \frac{1}{m}\sum_{i=1}^{m} \log \left[\sum_{j=1}^{k} \pi_j \mathcal{N}\left(\mathbf{x}^{(i)}; \mu_j, \Sigma_j\right)\right]$$

subject to the constraints that $\pi_j \geq 0$ and $\sum_{j=1}^{k} \pi_j = 1$.

1. **M-step:** Writing down the KKT necessary and sufficient optimality conditions (see (3.88) on page 242) for this maximization problem, subject to its associated inequality and linear equality constraints yields:

   (a) For μ_j

$$\mu_j = \frac{\sum_{i=1}^{m} p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)\mathbf{x}^{(i)}}{\sum_{i=1}^{m}\sum_{j=1}^{k} p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)} \tag{1.64}$$

   (b) And for Σ_j

$$\Sigma'_j = \frac{\sum_{i=1}^{m} p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)\left(\mathbf{x}^{(i)} - \mu_j\right)\left(\mathbf{x}^{(i)} - \mu_j\right)^T}{\sum_{i=1}^{m}\sum_{j=1}^{k} p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)} \tag{1.65}$$

   These steps are called the $M - steps$ or the maximization steps, since they are obtained as necessary and sufficient conditions of optimality for a maximization problem.

2. **E-step:** The posterior $p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)$ and the prior π_j in (1.65) and (1.64) can be determined using Bayes rule as

   (a)
$$p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma) = \frac{\pi_j f(\mathbf{x}; \mu_j, \Sigma_j)}{\sum_{a=1}^{k} \pi_a f(\mathbf{x}; \mu_a, \Sigma_a)}$$

   (b)
$$\pi_j = \frac{1}{m}\sum_{i=1}^{m} p(z_j \mid \mathbf{x}^{(i)}, \mu, \Sigma)$$

The problem is that we do not get a closed form solution here; what we obtain are a set of coupled, non-linear equations and need to iterate between these steps to arrive at the fix point. This is where the expectation maximization (EM) algoriothm comes in. We now will specify the EM algorithm in a more general setting.

1.8.2 Expectation Maximization

Let $\mathbf{X}$ be a set of observed variables and $\mathbf{Z}$ be a set of hidden variables for some statistical model. Let $\overline{\mathbf{x}}$ be m observations on $\mathbf{X}$. In this general setting, we really need not assume that the samples in $\overline{\mathbf{x}}$ are iid (though you could). We will assume that the MLE problem would have been easy[47] if $\overline{\mathbf{z}}$ was observed data for the hidden variables $\mathbf{Z}$ (such as in the case of the mixture model). The complete data log-likelihood would have been:

$$LL(\theta; \overline{\mathbf{x}}, \overline{\mathbf{z}}) = \frac{1}{m} \log p(\overline{\mathbf{x}}, \overline{\mathbf{z}}; \theta)$$

Given a predictive distribution $q(\mathbf{z}|\mathbf{x})$, the expected complete data log-likelihood is a function of the observed $\overline{\mathbf{x}}$ and θ and is defined as

$$LL_E(\theta; \overline{\mathbf{x}}) = \sum_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \log p(\overline{\mathbf{x}}, \overline{\mathbf{z}}; \theta) \tag{1.66}$$

The expected complete data log-likelihood is an auxilliary function that gives a lower bound on the actual log-likelihood we want to optimize for. The actual log-likelihood in this general setting will be:

$$LL(\theta; \overline{\mathbf{x}}) = \frac{1}{m} \log \left\{ \sum_{\mathbf{z}} p(\overline{\mathbf{x}}, \mathbf{z}; \theta) \right\}$$

For example, the actual log-likelihood with iid assumption will be:

$$LL(\theta; \overline{\mathbf{x}}) = \frac{1}{m} \sum_{i=1}^{m} \log \left\{ \sum_{\mathbf{z}} p(\mathbf{x}^{(i)}, \mathbf{z}; \theta) \right\}$$

Theorem 16 *For all θ and every possible distribution $q(\mathbf{z}|\mathbf{x})$, following holds:*

$$LL(\theta; \overline{\mathbf{x}}) \geq LL_E(\theta; \overline{\mathbf{x}}) + \frac{1}{m} H(q) \tag{1.67}$$

Equality holds if and only if

$$q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x}; \theta)$$

Proof: First of all

$$LL(\theta; \overline{\mathbf{x}}) = \frac{1}{m} \log \left\{ \sum_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \frac{p(\overline{\mathbf{x}}, \mathbf{z}; \theta)}{q(\mathbf{z}|\overline{\mathbf{x}})} \right\}$$

[47]The trick in such a setting is to identify the model, $\mathbf{X}$ and $\mathbf{Z}$ so that you make the MLE problem easy in the presence of complete data.

Using the Jensen's inequality (since *log* is a strictly convex function),

$$LL(\theta;\overline{\mathbf{x}}) \geq \underbrace{\frac{1}{m}\sum_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \log p(\overline{\mathbf{x}},\mathbf{z};\theta)}_{LL_E(\theta;\overline{\mathbf{x}})} \underbrace{-\frac{1}{m}\sum_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \log q(\mathbf{z}|\overline{\mathbf{x}})}_{H(q)}$$

Equality holds if and only if $\frac{p(\overline{\mathbf{x}},\mathbf{z};\theta)}{q(\mathbf{z}|\overline{\mathbf{x}})}$ is a constant, that is,

$$q(\mathbf{z}|\overline{\mathbf{x}}) \propto p(\overline{\mathbf{x}},\mathbf{z};\theta) = p(\mathbf{z}|\overline{\mathbf{x}};\theta)p(\overline{\mathbf{x}};\theta) \propto p(\mathbf{z}|\overline{\mathbf{x}};\theta)$$

This can happen if and only if $q(\mathbf{z}|\overline{\mathbf{x}}) = p(\mathbf{z}|\overline{\mathbf{x}};\theta)$. □

A consequence of theorem 16 is that

$$\max_{\theta} LL(\theta;\overline{\mathbf{x}}) = \max_{\theta} \max_{q} LL_E(\theta;\overline{\mathbf{x}}) + \frac{1}{m}H(q) \tag{1.68}$$

The EM algorithm is simply coordinate ascent on the auxilliary function $LL_E(\theta;\overline{\mathbf{x}}) + \frac{1}{m}H(q)$. The expectation and maximization steps at time instance t can be easily identified for the formulation in (1.68) as

1. **Expectation Step:**

$$q^{(t+1)} = \underset{q}{\operatorname{argmax}}\, LL_E(\theta^{(t)};\overline{\mathbf{x}}) + \frac{1}{m}H(q) = \underset{q}{\operatorname{argmax}} -D\left(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x};\theta^{(t)})\right) + \log\left\{\mathbf{x};\theta^{(t)}\right\} \tag{1.69}$$

Since, $LL_E(\theta^{(t)};\overline{\mathbf{x}}) + \frac{1}{m}H(q) \leq \log\left\{\mathbf{x};\theta^{(t)}\right\}$ by theorem 16, the maximum value is attained in (1.69) for $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x};\theta^{(t)})$. Thus, the E-step can be summarized by

$$q^{(t+1)}(\mathbf{x}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x};\theta^{(t)}) \tag{1.70}$$

The E-step can involve procedures such as sum-product for obtaining marginals and/or conditions, if the distribution is defined on a graphical model, to obtain $p(\mathbf{z}|\mathbf{x};\theta^{(t)})$.

2. **Maximization Step:**

$$\theta^{(t+1)} = \underset{\theta}{\operatorname{argmax}}\, LL_E(\theta;\overline{\mathbf{x}}) + \frac{1}{m}H(q^{(t+1)})$$

Since the maximization is over θ and since $H(q)$ is independent of θ, we can rewrite the M-step to be

$$\theta^{(t+1)} = \underset{\theta}{\text{argmax}}\ LL_E(\theta;\overline{\mathbf{x}}) = \underset{\theta}{\text{argmax}}\ \sum_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \log p(\overline{\mathbf{x}},\overline{\mathbf{z}};\theta) \qquad (1.71)$$

In essence, the M-step looks very much like an ordinary maximum likelihood estimation problem, but using predicted values of $\mathbf{z}$. The M-step may not have a closed form solution, in which case, it may be required to resort to iterative techniques such as IPF (1.7.6).

Let us take some examples to illustrate the generic EM procedure outlined here. It is particularly useful if the term $\log p(\overline{\mathbf{x}},\overline{\mathbf{z}};\theta)$ were to split up into smaller terms (such as sum of sufficient statistics in the case of exponential models). Consider the Gauss Markov process specified by $Z_{t+1} = \theta Z_T + W_t$, where $Z_0 \sim \mathcal{N}(0,1)$ and $W_t \sim \mathcal{N}(0,1)$. Let $\theta \in \Re$ be the parameter to be estimated. The graphical model representation is $\mathcal{V} = \{Z_1, Z_2, Z_2, \ldots, Z_n, X_1, X_2, \ldots, X_n\}$ and $\mathcal{E} = \{(Z_1,Z_2),(Z_1,X_1),(Z_2,Z_3),(Z_2,Z_2),\ldots,(Z_{n-1},Z_n),(Z_{n-1},X_{n-1}),(Z_n,X_n)\}$.

Say what we observe are noisy, indirect observations $X_t = Z_t + V_t$, $V_t \sim \mathcal{N}(0,\sigma_2)$ being the observation noise. Let $\overline{\mathbf{x}}$ be a set of m iid observations for $\mathbf{X}$ while $\mathbf{Z}$ remains hidden. Both $\mathbf{X}$ and $\mathbf{Z}$ are vectors of random variables of size n each. Then,

$$\begin{aligned} LL(\theta;\overline{\mathbf{x}}) &= \frac{1}{m}\sum_{i=1}^{m} p(x^{(i)};\theta) \\ &= \frac{1}{m}\sum_{i=1}^{m} \log\left\{\int_{\mathbf{z}} \prod_{t=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\frac{(x_t^{(i)}-z_t)^2}{\sigma^2} p(\mathbf{z};\theta)d\mathbf{z}\right\}\right\} \quad (1.72) \end{aligned}$$

which is a mess! In contrast, the lower-bound component LL_E allows us to move the integration outside the logarithm, enabling more simplification:

$$\begin{aligned} LL_E(\theta;\overline{\mathbf{x}}) &= \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \log p(\overline{\mathbf{x}}\mathbf{z};\theta)d\mathbf{z} \\ &= \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \sum_{i=1}^{m} \log p(\mathbf{x}^{(i)}\mathbf{z};\theta)d\mathbf{z} \\ &= \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}}) \sum_{i=1}^{m} \left[\log p(\mathbf{z},\theta) + \sum_{t=1}^{n}\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right) - \frac{1}{2\sigma^2}(x_t^{(i)}-z_t)^2\right] d\mathbf{z} \end{aligned}$$

As can be seen above, $p(\mathbf{x}^{(i)}|\mathbf{z};\theta)$ is independent of θ and therefore, the term $q(\mathbf{z}|\overline{\mathbf{x}}) \log p(\mathbf{x}^{(i)}|\mathbf{z};\theta)$ can be brought out as a separate constant (since that part of the integral will not change with θ). This leads to the following simplified expression for LL_E

$$\begin{aligned}
LL_E(\theta;\overline{\mathbf{x}}) &= \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}})\log p(\overline{\mathbf{x}}\mathbf{z};\theta)d\mathbf{z} \\
&= C + \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}})\left[\log p(z_1) + \log p(z_2|z_1;\theta) + \ldots + \log p(z_n|z_{n-1};\theta)\right] d\mathbf{z} \\
&= C' + \frac{1}{m}\int_{\mathbf{z}} q(\mathbf{z}|\overline{\mathbf{x}})\left[\log p(z_2|z_1;\theta) + \ldots + \log p(z_n|z_{n-1};\theta)\right] d\mathbf{z}
\end{aligned}$$

In the E-step, the Kalman filter can be used to compute $p(\mathbf{z}|\mathbf{x};\theta^{(t+1)})$ in terms of $\theta^{(t)}$. In the M-step, first order necessary optimality conditions on LL_E will yield $\theta^{(t+1)}$.

Recall from Section 1.7.7, equation (1.63) that the likelihood maximization problem can be viewed as a problem of minimizing the KL divergence between the empirical distribution and the model distribution.

$$\widehat{p}_{MLE} = \underset{p\in\mathbf{E}(\mathbf{f})}{\mathrm{argmin}} D\left(\wp_{\overline{\mathbf{x}}}||p(\mathbf{x};\theta)\right)$$

While the likelihood maximization perspective lead to a lower-bounding strategy in the form of EM, an alternative upper-bounding strategy can also be adopted to view EM, though it is only the older bound in disguise. Making use of theorem 16, we can prove that for all distributios $q(\mathbf{z}|\mathbf{x})$ and any parameter θ, the following always holds:

$$D\left(\wp(\mathbf{x})||p(\mathbf{x};\theta)\right) \le D\left(\wp(\mathbf{x})q(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z};\theta)\right) \qquad (1.73)$$

This statement says that the KL divergence between the empirical and model distributions that maximum likelihood tries to minimize is upperbounded by the KL divergence between the 'completed' empirical and model distributions. As before, it can also be shown that the bound is tight if and only if $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x};\theta)$. The E-step will remain the same as before. Only, the M-step will slightly change:

1. **KL divergence perspective of E-Step:**

$$q^{(t+1)}(\mathbf{z}|\mathbf{x}) = \underset{q}{\mathrm{argmin}} D\left(\wp(\mathbf{x})q(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z};\theta^{(t)})\right)$$

2. **KL divergence perspective of M-Step:**

$$\theta^{(t+1)} = \underset{\theta}{\mathrm{argmin}} D\left(\wp(\mathbf{x})q^{(t+1)}(\mathbf{z}|\mathbf{x})||p(\mathbf{x},\mathbf{z};\theta)\right)$$

These modified E and M steps correspond to coordinate descent in constrast to the earlier perspective of coordinate ascent.

1.9 Variational Methods for Inference

In contrast to the sampling methods, variational methods are deterministic and fast algorithms that generate good approximations to the problems of computing marginals and MAP configurations. They are involve the reformulation of the quantity of interest (such as log-likelihood, first order moments, marginal distributions, *etc.*) as the solution of some optimization problem. They are useful in several ways:

- The variational formulation often leads to efficient algorithms for determining exact solutions. Many algorithms discussed thus far, could be discovered as efficient techniques for solving the variational optimization problem.
- For many quantities that are hard to compute, the variational perspective leads to approximate solutions. Mean field and loopy sum product algorithms can also be viewed as special cases of approximation through variational inference.
- In contrast to approximate sampling methods, these are faster, deterministic and cheaper (in terms of memory).

We will motivate variational methods using two examples.

1. The first is the mean field algorithm for the Ising model defined on a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$ of binary (0/1) variables, with pairwise interactions between adjacent variables captured through their product $(x_s x_t)$

$$p(\mathbf{x}, \eta) \propto \exp\left\{ \sum_{s \in \mathcal{V}} \eta_s x_s + \sum_{(s,t) \in \mathcal{E}} \eta_{st} x_s x_t \right\}$$

The Gibbs sampling for the Ising model derives updates of the form

$$x_i^{(t+1)} = \begin{cases} 1 & \text{if } u \sim uniform[0,1] \leq 1 + \exp\left\{ -\eta_i + \sum_{j \in \mathcal{N}(s)} \eta_{ij} x_j^{(t)} \right\} \\ 0 & \text{otherwise} \end{cases}$$

which correspond to a non-deterministic version of the message passing algorithm (owing to $u \sim uniform[0,1]$). The updates are very simple and local, making this a good choice.

The mean field method has its roots in physics. It makes a deterministic to the Gibbs update by replacing each random variable X_i by a deterministic mean parameter $\mu_i \in [0,1]$ (which can be thought of as the probability of $X_i = 1$) and updates μ_i using

$$\mu_i^{(t+1)} = \left(1 + \exp\left\{ -\eta_i + \sum_{j \in \mathcal{N}(s)} \eta_{ij} \mu_j^{(t)} \right\} \right)^{-1}$$

Thus, the mean field algorithm is exactly a message passing algorithm, but has some semantics related to sampling techniques. We will see that the mean field algorithm is a specific instance of variational methods and it can be formulated as coordinate descent on a certain optimization problem and subsequently be analysed for convergence, *etc.*

2. The *loopy sum-product* (also called the loopy belief propagation) method is another instance of variational methods. 'Loopy' here means on graphs with cycles. If the tree width were low, you could create a junction tree and perform message passing, but what if the tree width were large, such as with a grid. This algorithm is the most naive application of the sum-product updates (originally developed for trees in Section 1.2.2) and apply it to graphs with cycles. This naive procedure has had extraordinary success in the fields of signal processing, compute vision, bioinformatics and most importantly, in communication networks, *etc.*

   The message passing algorithm, when applied on a tree, breaks it into subtrees and passes messages between subtrees, which are independent (share no variable). But the moment you add an edge connecting any two subtrees, they are not independent any more. Passing messages around in cycles can lead to over-counting (analogous to gossip spreading in a social network). Thus, the message passing algorithm ceases to remain an exact algorithm and does not even gurantee convergence.

   What turns out to be actually very important is how long are the cycles on which messages are being propagated; for *long cycles, the effects of over-counting can be weakened.* More technically speaking, the behaviour will depend on

   (a) The girth of the graph (length of cycles): For larger girth, you could run the message passing for many iterations before you land up with overcounting.

   (b) The strength of the potentials are, or in other words, how close to independence is the model. For example, in the Ising model itself, based on the coupling induced through terms of the form $x_s x_t$, if the coupling is weak, almost close to independence, the algorithm will be perfect, giving almost exact answers. There is a region of transition, based on strengthening of the coupling terms, beyond which the algorithm breaks down.

3. The idea behind variational methods is to represent the quantity of interest (such as the marginal or mode over a graphical model) as the solution to an optimization problem. For example, the solution to $A\mathbf{x} = \mathbf{b}$ (as in the case of inferencing for Kalman filters) with $A \succ 0$ and $\mathbf{b} \in \Re^n$ can be represented as the solution to

$$\widetilde{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

This is precisely a *variational formulation* for the linear system $A\mathbf{x} = \mathbf{b}$. If the system of equations $A\mathbf{x} = \mathbf{b}$ is large[48] the solution $\widetilde{\mathbf{x}} = A^{-1}\mathbf{b}$ may not be easy to compute, in the event of which, iterative (and sometimes approximate) solutions to the optimization problem can be helpful. One of the most succesful techniques for solving such systems (without inverting matrices) is the conjugate gradient method, discussed in Section 3.5.8, that solves the system in exactly $O(n)$ steps.

4. As will be seen on page 48 in Section 1.4, the bernoulli distribution can be expressed as

$$p(\mathbf{x}, \eta) = \exp\{\eta x - A(\eta)\}$$

for $X \in \{0, 1\}$. $A(\eta) = \log(1 + e^{\eta})$. We saw in Section 1.4 that the mean is given by

$$\overline{\mu} = E_{\eta} = \nabla A(\eta) = \frac{e^{\eta}}{1 + e^{\eta}} = (1 + e^{-\eta})^{-1}$$

The key is in noting that $\overline{\mu}$ corresponds to the 'slope of' a supporting hyperplane (see Figure 3.38 on page 217) for $epi(A(\eta))$ in a $(\eta, A(\eta))$ space. Thus, we are interested in all the hyperplanes that lie below $epi(A(\eta))$, with intercept C along the axis for $A(\eta)$:

$$\mu^T\eta - C \leq A(\eta)$$

and want to get as close to a supporting hyperplane as possible

$$C^* = \sup_{\eta} \{\mu^T\eta - A(\eta)\} = A^*(\mu)$$

Borrowing ideas from duality theory (*c.f.* Section 3.4), we call the function $\sup_{\eta} \{\mu^T\eta - A(\eta)\}$ as the dual function $A^*(\mu)$. C^* is the intercept for the supporting hyerplane. In the case of the bernoulli example, we have

$$A^*(\mu) = \sup_{\eta} \{\mu^T\eta - \log 1 + e^{\eta}\} = \begin{cases} (1-\mu)\log(1-\mu) + \mu\log\mu & \text{if } \mu \in (0,1) \\ \infty & \text{otherwise} \end{cases}$$

Under nice conditions (such as under *Slaters constraint qualification* discussed in definition 52 on page 236), the operation of taking duals is symmetric $((A^*)^* = A)$, that is,

$$A(\theta) = \sup_{\mu} \{\mu^T\eta - A^*(\mu)\}$$

Under the conditions of zero duality gap, it should happen that if

$$\widehat{\mu}(\eta) = \underset{\mu}{\operatorname{argmax}} \{\mu^T\eta - A^*(\mu)\}$$

[48]Of course, as we saw in Section 1.5, such a system comes up in the case of solution to Kalman filters, but can be computed efficiently by exploiting the tree structure of the graph in inverting the matrix. The discussion here is for general graphs.

is the primal optimum, then $\widehat{\mu}^T\eta - A^*(\widehat{\mu})$ is the supporting hyperplane to $A(\eta)$, meaning that $\widehat{\mu}$ is the mean we were seeking. This yields a variational representatation for the original problem of finding the mean. The dual itself is also useful in determining the log-normalization constant for problems such as parameter estimation. We can confirm in the simple bernoulli case that indeed

$$\widehat{\mu}(\eta) = \overline{\mu}(\eta) = \frac{e^\eta}{1+e^\eta}$$

We will now generalize the variational formulation of the bernoulli case to the exponential family.

$$p(\mathbf{x};\theta) = \exp\left\{\theta^T\mathbf{f}(\mathbf{x}) - A(\theta)\right\}$$

where, $\mathbf{x} \in \Re^n$ and $\mathbf{f} : \Re^n \rightarrow \Re^d$. Let us say we are interested in computing the first order moments

$$\mu = E[\mathbf{f}(\mathbf{x})] \tag{1.74}$$

Following a similar line of argument as for the case of the bernoulli distribution, we define the dual as

$$A^*(\mu) = \sup_{\theta}\ \left\{\mu^T\theta - A(\mu)\right\}$$

The key ingredients in this calculation are to set the gradient with respect to θ to $\mathbf{0}$, as a necessary first order condition.

$$\mu - \nabla A(\theta) = \mathbf{0} \tag{1.75}$$

This looks like the moment matching problem that results from maximum likelihood estimation. The only difference is that μ need not come from data, it is the argument to $A^*(\eta)$. When will (1.75) have a solution? In the simple bernoulli case, we already saw that we will have a solution $\eta = -\log\mu - \log 1 - \mu$ if $\mu \in (0,1)$. As another example, for the univariate Gaussian, $\mathbf{f}(\mathbf{x}) = [x, x^2]$ and $A(\eta) = \frac{1}{2\sigma^2}\mu^2 + \log\sigma \equiv -\frac{\eta_1^2}{4\eta_2} + \frac{1}{2}\log(-2\eta_2)$ (as seen on page 48). The system (1.75) can be shown to have a solution only if $\mu_2 - \mu_1^2 > 0$, that is if the variance is positive. For two examples, the conditions under which solutions exist to (1.75) are extremely simple - it should be possible to generate data using the distribution.

Assuming that a solution $\theta(\mu)$ exists to (1.75), and exploiting the fact that $\theta(\mu)$ satisfies moment matching conditions (1.75)

$$\sum_{\mathbf{x}} p(\mathbf{x};\theta(\mu))\mathbf{f}(\mathbf{x}) = \mu \tag{1.76}$$

and using the property that

$$A(\mu) = \sum_{\mathbf{x}} A(\mu) p(\mathbf{x};\theta(\mu))$$

the dual will have the form

$$\begin{aligned}
A^*(\mu) &= \theta^T(\mu)\mu - A(\mu) \\
&= \theta^T(\mu)\left(\sum_{\mathbf{x}} p(\mathbf{x};\theta(\mu))\mathbf{f}(\mathbf{x})\right) - A(\mu) \\
&= \sum_{\mathbf{x}} p(\mathbf{x};\theta(\mu))\left(\theta^T(\mu)\mathbf{f}(\mathbf{x}) - A(\mu)\right) \\
&= \sum_{\mathbf{x}} p(\mathbf{x};\theta(\mu)) \log p(\mathbf{x};\theta(\mu)) \\
&= -H(p(\mathbf{x};\theta(\mu))) \qquad (1.77)
\end{aligned}$$

That is, the dual is precisely the negative entropy $-H(p(\mathbf{x};\theta(\mu)))$ of the distribution whose parameters $\theta(\mu)$ are obtained by moment matching. The dual for the bernoulli case which resembled an entropy was by no means a coincidence! If $\mathcal{M}$ is the set of all possible moments that make a solution to the system (1.75 or equivalently 1.76) feasible, that is

$$\mathcal{M} = \left\{ \mu \in \Re^d \,\middle|\, \sum_{\mathbf{x}} p(\mathbf{x};\theta)\mathbf{f}(\mathbf{x}) = \mu \; for \; some \; p(.) \right\}$$

then the dual could also be expressed as

$$A^*(\mu) = \begin{cases} -\max \; H(p(\mathbf{x};\theta(\mu))) & \text{such that } E[\mathbf{f}(\mathbf{x})] = \mu \text{ for } \mu \in \mathcal{M} \\ \infty & \text{otherwise} \end{cases}$$

Another way of characterizing $\mathcal{M}$ is as the set of all first order moments that could be generated by $p(\mathbf{x};\theta)$. In any case, $\mathcal{M}$ is often very had to characterize and loop belief propagation *etc.* are often used to approximate it.

Finally, we will write down the variational problem as a reformulation of (1.74), of which mean field, (loopy) sum-product, Gauss Seidel, Jacobi, etc can be found to be special cases. Writing down the dual of the dual of (1.77), and assuming zero duality gap, we get a reformulation of (1.74):

$$A(\theta) = \sup_{\mu \in \mathcal{M}} \left\{ \mu^T\theta - A^*(\mu) \right\} \qquad (1.78)$$

Again, $A(\theta)$ is a very hard function to compute, mainly because $\mathcal{M}$ is simple to characterize. This maximumization problem is concave, since $A^*(\eta)$ is concave and the constraint set $\mathcal{M}$ is convex. Under zero duality gap conditions (which holds in this case), the optimal point will be achieved at $\widehat{\mu}(\theta) = E[\mathbf{f}(\mathbf{x})]$.

We will us take some examples to illustate the use of variational techniques. For problems of estimating moments, **f** could be the feature functions. For problems of estimating marginals, **f** can be chosen to be the indicator function.

The simplest example is for a two node chain: $\mathcal{V} = \{X_1, X_2\}$, $\mathcal{E} = \{(X_1, X_2)\}$, $X_1, X_2 \in \{0, 1\}$ and

$$p(\mathbf{x}; \theta) \propto \exp\{\theta_1 x_1 + \theta_2 x_2 + \theta_{12} x_1 x_2\}$$

The moments are: $\mu_i = E[X_i] = p(X_i = 1)$ and $\mu_{12} = E[X_1 X_2] = p(X_1 = 1, X_2 = 1)$. The set $\mathcal{M}$ is

$$\mathcal{M} = \left\{ \mu \in \Re^3 \left| \sum_{\mathbf{x}} p(\mathbf{x}; \theta)\mathbf{f}(\mathbf{x}) = \mu \ for\ some\ p(.) \right. \right\} = \{\mu_i \in [0, 1],\ 0 \le \mu_{12} \le \min(\mu_1, \mu_2),\ 1 + \mu_{12} - \mu_1 - \mu_2 \ge 0\}$$

Let us next write down the dual in terms of the entropy of the distribution

$$\begin{aligned} A^*(\mu) = -H(p(\mathbf{x}; \mu)) &= \sum_{x_1, x_2} p(x_1, x_2) \log p(x_1, x_2) \\ &= \mu_{12} \log \mu_{12} + (\mu_1 - \mu_{12}) \log (\mu_1 - \mu_{12}) + (\mu_2 - \mu_{12}) \log (\mu_2 - \mu_{12}) \\ &+ (1 + \mu_{12} - \mu_1 - \mu_2) \log (1 + \mu_{12} - \mu_1 - \mu_2) \end{aligned} \tag{1.79}$$

The corresponding variational problem will be

$$A(\theta) = \max_{\mu \in \mathcal{M}} \{\theta_1 \mu_1 + \theta_2 \mu_2 + \theta_{12} \mu_{12} - A^*(\mu)\}$$

Though this can be solved using the method of Lagrange multipliers (*c.f.*, Section 3.4.1), *etc.*, we expect the optimal solution to the variational problem to be

$$\widehat{\mu}_1 = \frac{1}{z} \sum_{x_1 \in \{0,1\}, x_2 \in \{0,1\}} x_1 \exp\{\theta_1 x_1 + \theta_2 x_2 + \theta_{12} x_1 x_2\} = \frac{\exp \theta_1 + \exp \theta_1 + \theta_2 + \theta_{12}}{1 + \exp \theta_1 + \exp \theta_2 + \exp \theta_1 + \theta_2 + \theta_{12}}$$

There are many applications of variational inference to quantity estimation problems that have either no exactly solutions, or that have solutions not computable in polynomial time. Further, variational principles can be used to study how the approximate algorithms behave; whether they have fix points, whether they converge. what answers do they give, *etc.*. For instance, in belief propagation from a variational perspective, the messages correspond to lagrange multipliers (for active constraints in $\mathcal{M}$) that are passed around.

In general, the sets $\mathcal{M}$ are very complex. For example, with a complete 7 node graph, there will be $O(3 \times 10^8)$ constraints. For an n node tree, you will have $4(n - 1)$ constraints. The variational principle provides the foundation for many approximate methods such as the Naive mean field algorithm, which restricts optimization to a 'tractable' set of $\mathcal{M}$, such as one for which the joint distribution over a graphical model is factorized by treating the variables as completely independent.

Chapter 2

Linear Algebra

Dixit algorizmi. Or, "So said al-Khwarizmi", being the opening words of a 12$^{\text{th}}$ century Latin translation of a work on arithmetic by al-Khwarizmi (*ca.* 780–840).

2.1 Linear Equations

Elementary algebra, using the rules of completion and balancing developed by al-Khwarizmi, allows us to determine the value of an unknown variable x that satisfies an equation like the one below:

$$10x - 5 = 15 + 5x$$

An equation like this that only involves an unknown (like x) and not its higher powers (x^2, x^3), along with additions (or subtractions) of the unknown multiplied by numbers (like $10x$ and $5x$) is called a *linear* equation. We now know, of course, that the equation above can be converted to a special form ("number multiplied by unknown equals number", or $ax = b$, where a and b are numbers):

$$5x = 20$$

Once in this form, it becomes easy to see that $x = b/a = 4$. Linear algebra is, in essence, concerned with the solution of several linear equations in several unknowns. Here is a simple example of two equations and two unknowns x and y, written in a uniform way, with all unknowns (variables) to the left of the equality, and all numbers (constants) to the right:

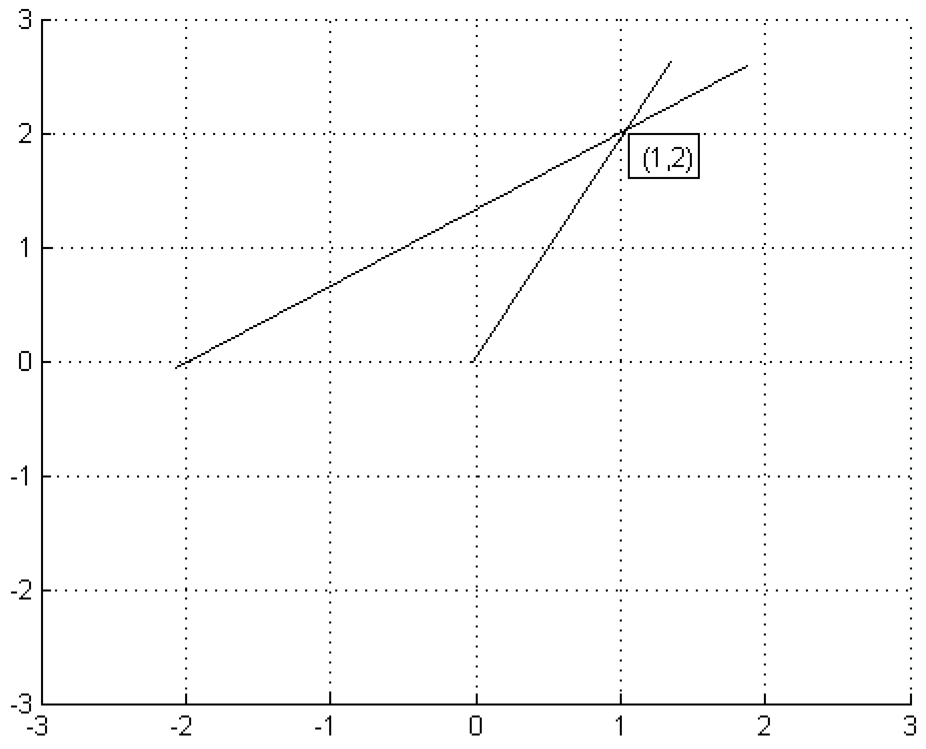


Figure 2.1: Solving linear equations: the geometric view.

$$2x - y = 0$$

$$-x + 2y = 3$$

We woud like to find values of x and y for which these equations are true. School geometry tells us how to visualise this: each equation is a straight line in the xy plane, and since we want a value of x and y for which both equations are true, we are really asking for the values of x and y that lie on both lines (that is, the point of intersection of the two lines: see Fig. 2.1). Of course, if the lines do not meet at a point, then there are no values of x and y that satisfy the equations. And we can continue to solve problems like these geometrically: more unknowns means lines become higher-dimensional flat surfaces ("hyperplanes"), and more equations means we are looking for the single point of intersection of all these surfaces. Visually though, this is challenging for all but a small minority of us, geared as we are to live in a world of three spatial dimensions.

Linear algebra, an extension of elementary algebra, gives us a way of looking at the solution of any number of linear equations, with any number of variables without suffering from this visual overload. In effect, equations are once again converted to the simple form we just saw, that is, $Ax = b$, although A and b are no longer just numbers. In fact, we will see that A is a *matrix*, and that x and b are *vectors* (and in order not to confuse them with variables and numbers, we will from now on use the bold-face notation $\mathbf{x}$ and $\mathbf{b}$). Linear algebra, shows us that solutions, if they exist, can be obtained in three different ways:

1. A direct solution, using techniques called elimination and back substitution.
2. A solution by "inverting" the matrix A, to give the solution $\mathbf{x} = A^{-1}\mathbf{b}$.
3. A vector space solution, by looking at notions called the column space and nullspace of A.

Understanding each of these requires a minimal understanding of vectors and matrices, which we give in a somewhat compressed form here.

2.2 Vectors and Matrices

It is easiest to think of a vector as a generalisation of a single number. A pair of numbers can be represented by a *two-dimensional vector*. Here is the two-dimensional vector representation of the pair $(2, -1)$:

$$\mathbf{u} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

This kind of vector is usually called a "column" vector. Geometrically, such a vector is often visualised by an arrow in the two-dimensional plane as shown on the left in Fig. **??**. Multiplying such a vector by any particular number, say 2, multiplies each component by that number. That is, $2\mathbf{u}$ represents the pair $(4, -2)$. Geometrically, we can see that multiplication by a number—sometimes called *scalar multiplication*—simply makes gives a vector with a "longer" arrow as shown on the right in the figure (assuming, of course, that we are not dealing with zero-length vectors). In general, multiplication of a (non-zero) vector $\mathbf{u}$ by different (non-zero) numbers a result in lines either in the direction of $\mathbf{u}$ (if $a > 0$) or in the opposite direction
Suppose we now consider a second vector $\mathbf{v}$ corresponding to the pair $(-1, 2)$, and ask: what is $\mathbf{u} + \mathbf{v}$. This simply adds the individual components. In our example:

$$\mathbf{u} = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \mathbf{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix} \quad \mathbf{u} + \mathbf{v} = \begin{bmatrix} 2-1 \\ -1+2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Geometrically, the addition of two vectors gives a third, which can visualised as the diagonal of the parallelogram formed by **u** and **v** (Fig. **??**, left). It should be straightforward to visualise that any point on the plane containing the vectors **u** and **v** can be obtained by some linear combination $a\mathbf{u} + b\mathbf{v}$, and that the space of all linear combinations is simply the full two-dimensional plane containing **u** and **v** (Fig. **??**, right). For the two-dimensional example here, this plane is just the usual xy plane (we will see that this is the vector space $\Re^2$).
Although we have so far only looked at vectors with two components, linear algebra is more general. It allows us to use the same operations with vectors of any size. Suppose our vectors **u** and **v** are three-dimensional. Linear combinations now still fill a plane containing the two vectors. But, this is no longer the xy plane, since the vectors generated by the linear combinations are points in three-dimensional space (we will see later, that is some "subspace" of the vector space $\Re^3$). Addition of a third vector **w** will also not necessarily result in a point on this plane, and the space of linear combinations $a\mathbf{u} + b\mathbf{v} + c\mathbf{w}$ could fill the entire three-dimensional space.

Let us return now to the two equations that we saw in the previous section:

$$2x - y = 0$$
$$-x + 2y = 3$$

It should be easy to see how these can be written in "vector" form:

$$x \begin{bmatrix} 2 \\ -1 \end{bmatrix} + y \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \tag{2.1}$$

That is, we are asking if there is some linear combination of the column vectors $[2, -1]$ and $[-1, 2]$ that gives the column vector $[0, 3]$. And this is the point of departure with the usual geometric approach: we visualise solutions of equations not as points of intersections of surfaces, but as linear combination of vectors (of whatever dimension): see Fig. 2.2.

To get it into a form that is even more manageable, we need the concept of a "coefficient matrix". A matrix is simply a rectangular array of numbers, and the coefficient matrix for the left hand side of the linear combination above is:

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

This is a 2×2 ("two by two") matrix, meaning it has 2 rows and 2 columns. You can see that the columns of the matrix are simply the column vectors of the linear combination. Let:

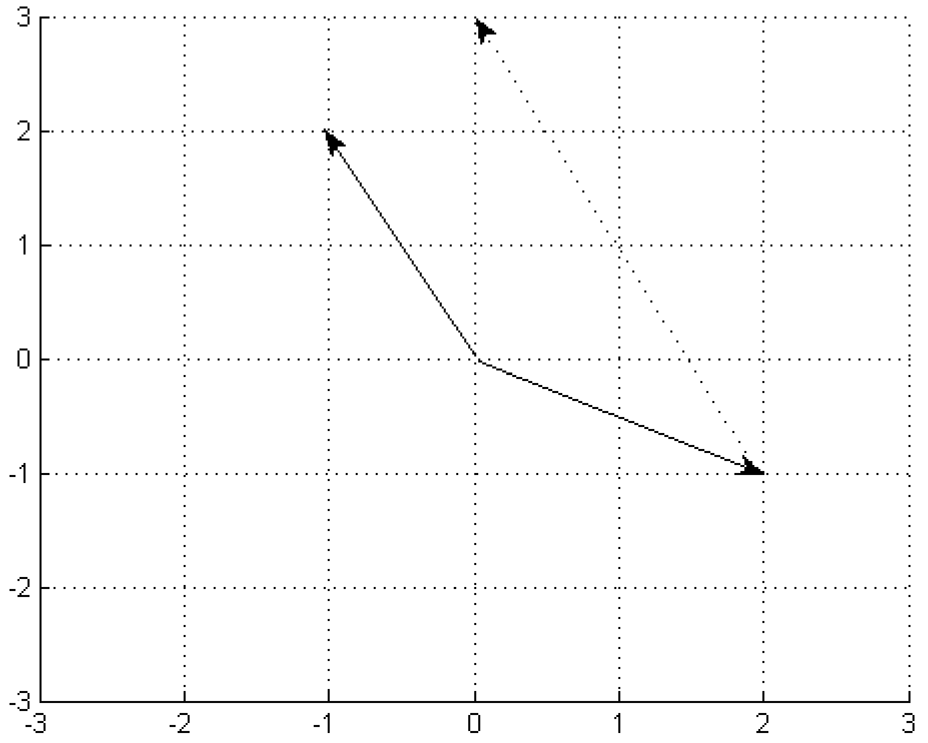


Figure 2.2: Solving linear equations: the geometric view from linear algebra.

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$$

Then, the matrix equation representing the same linear combination is:

$$A\mathbf{x} = \mathbf{b} \tag{2.2}$$

This, as you can see, is just as simple, at least in form. as the very first equation we started with ($5x = 20$). We still need to know what $A\mathbf{x}$ means. Comparing Equations 2.2 and 2.2, $A\mathbf{x} = x$ (column 1 of A) + y (column 2 of A).

This extends easily enough to equations with more variables. Here are three linear equations in three unknowns:

$$2x - y = 0$$
$$-x + 2y - z = -1$$
$$-3y + 4z = 4$$

The coefficient matrix A is:

$$A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -3 & 4 \end{bmatrix}$$

The right hand side of the matrix equation is:

$$\mathbf{b} = \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}$$

What we are trying to do is to find values of x, y and z such that:

$$x(\text{column 1 of } A) + y(\text{column 2 of } A) + z(\text{column 3 of } A) = \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}$$

It is easy to see now that the solution we are after is $x = 0, y = 0, z = 1$. Or, in vector form, the solution to the matrix equation $A\mathbf{x} = b$ is:

$$\mathbf{x} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

In general, things are not so obvious and it may be the case that for some values of A and b, no values of x, y and z would solve $A\mathbf{x} = b$. For example, $\mathbf{b}$ may be a point in 3-dimensional space that could not be "reached" by any linear combinations of the vectors comprising the columns of A. Here is a simple example:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Seqences of mathematical operations—algorithms, if you will—have been devised to check if solutions exist, and obtain these solutions mechanically when they exist. There are three such approaches we will look at: obtaining solutions by elimination (the simplest), obtaining solutions by matrix inversion (a bit more complex), and finally, a vector space solution (the hardest). We look at each of these in turn.

2.3 Solution of Linear Equations by Elimination

We will now examine a systematic method as "elimination"—first presented by Gauss, for solving a linear system of equations. The basic idea is to progressively *eliminate* variables from equations. For example, let us look once again at the two equations we saw earlier:

$$2x - y = 0$$

$$-x + 2y = 3$$

Elimination first multiplies both sides of the second equation by 2 (this clearly leaves it unchanged):

$$-2x + 4y = 6$$

We can also add equal amounts to the left and right sides without changing the equation. So, adding the left hand side of the first equation to the left hand side of this new equation, and the right hand side of the first equation to the right hand side of this new equation also does not alter anything:

$$(-2x+4y)+(2x-y)=6+0 \text{ or } 3y=6$$

So, the two original equations are the same as:

$$\begin{aligned} 2x-y &= 0 \\ 3y &= 6 \end{aligned}$$

You can see that x has been "eliminated" from the second equation and the set of equations have been said to be transformed into an *upper triangular* form. In this form, it is easy to see that $y = 6/3 = 2$. The value of x can then be obtained by substituting back this value for y in the first equation, to give $2x - 2 = 0$ or $x = 1$. The different steps in the elimination process can be expressed clearly using matrices, which we do now. As a running example, we will use the following set of 3 equations:

$$x+2y+z=2$$

$$3x+8y+z=12$$

$$4y+z=2$$

We now know what the coefficient matrix for these equations is:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix}$$

A point of notation. The entry in row 1, column 1 of A will be denoted a_{11}; row 1, column 2 will be a_{12} and so on. So, in the matrix above, $a_{11} = 1$, $a_{12} = 2$ *etc.*. In general, the entry in row i, column j will be denoted a_{ij}.

Before we plunge into the details of the matrix operations, let us just go through the procedure mechanically (taking on faith for the moment that the steps are indeed valid ones). Our first elimination step is to eliminate x from the second equation. We multiply the first equation by a multiplier and subtract it from the second equation with the goal of eliminating the x coefficient in the second equation. We will call it the $(2, 1)$ step. The first element of the first row a_{11} determines the value of the multiplier (3 in this case) and it is called a *pivot*. For reasons that will become clear, pivots should not be 0. The resultant coefficient matrix is:

$$A_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix}$$

The next step will be to get a 0 in the first column of the third row (a_{31}) of A_1. Since this is already the case, we do not really need to do anything. But, just to be pedantic, let us take it as giving a coefficient matrix A_2, which is just the same as A_1:

$$A_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix}$$

We now move on to eliminating a_{32} in A_2. Using a_{22} in A_2 as the next pivot, we subtract from the third row a multiple (2) of the second row. The resultant coefficient matrix is now:

$$A_3 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}$$

A_3 is called an upper triangular matrix for obvious reasons (and sometimes denoted by U). We will see shortly that with the sequence of operations that we have just done, the left hand side of the original matrix equation $A\mathbf{x}$ is transformed into $A_3\mathbf{x}$ by progressively multiplying by a sequence of matrices called "elimination matrices".

2.3.1 Elimination as Matrix Multiplication

Let us go back to the original matrix equation:

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix}$$

We take a step back and look again at the first elimination step ("multiply equation 1 by 3 and subtract from equation 2"). The effect of this step is to change the right-hand side second equation from 12 to $12 - 3 \times 2 = 6$ and leave the right-hand sides of all other equations unchanged. In matrix notation, the right hand side, after the first elimination step, is:

$$\mathbf{b}_1 = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}$$

A little calculation should be sufficient to convince yourself that $\mathbf{b}_1$ can be obtained by pre-multiplying $\mathbf{b}$ by the matrix:

$$E = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

That is, $\mathbf{b}_1 = E\mathbf{b}$. You can check this by doing the usual linear combination of the columns of E with the components of $\mathbf{b}$, but the following "row-by-column" view—which is simply the linear combination expanded out—may be even more helpful:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} a_{11}b_1 + a_{12}b_2 + a_{13}b_3 \\ a_{21}b_1 + a_{22}b_2 + a_{23}b_3 \\ a_{31}b_1 + a_{32}b_2 + a_{33}b_3 \end{bmatrix}$$

So, if the elimination step multiplies the left-hand side of of the matrix equation $A\mathbf{x} = \mathbf{b}$ by the matrix E, then to make sure nothing is changed, we have to do the same to the left-hand side. That is, the elimination step changes the left-hand side to $EA\mathbf{x}$. But now we are stuck—EA is a product of two matrices, which we have not come across before. What does this mean?

Well, we know what we would like EA to mean. We would like $EA = A_1$. That is:

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix} \tag{2.3}$$

Taking a vector as simply being a matrix with a single column, we would like to extend the old matrix-vector multiplication ($A\mathbf{x}$) idea to general matrix-matrix multiplication. Suppose B is a matrix comprised of the column vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$. Then AB is a matrix that has columns $A\mathbf{b}_1$, $A\mathbf{b}_2$, and $A\mathbf{b}_3$. So, in the example above, EA is a matrix that has columns $E\mathbf{a}_1$, $E\mathbf{a}_2$ and $E\mathbf{a}_3$ (where $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$ are the columns of A). Let us work out what these are:

$$E\mathbf{a}_1 = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \times 1 + 0 \times 3 + 0 \times 0 \\ -3 \times 1 + 1 \times 3 + 0 \times 0 \\ 0 \times 1 + 0 \times 3 + 1 \times 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

This is the first column of the matrix A_1 on the right-hand side of Equation 2.3.1. You can check that $E\mathbf{a}_2$ and $E\mathbf{a}_3$ do indeed give the other two columns of A_1. Once again, there is a "row-by-column" view of multiplying two matrices that you will often find helpful:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} =$$

$$\begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} & a_{11}b_{13} + a_{12}b_{23} + a_{13}b_{33} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} & a_{21}b_{13} + a_{22}b_{23} + a_{23}b_{33} \\ a_{31}b_{11} + a_{32}b_{21} + a_{33}b_{31} & a_{31}b_{12} + a_{32}b_{22} + a_{33}b_{32} & a_{31}b_{13} + a_{32}b_{23} + a_{33}b_{33} \end{bmatrix}$$

At this point, it is important that you are aware of some properties of matrix multiplication. First, multiplying matrices A and B is only meaningful if the number of columns of A is the same as the number of rows of B. If A is an $m \times n$ matrix, and B is an $n \times k$ matrix, then AB is an $m \times k$ matrix. Second, just like with ordinary numbers, matrix multiplication is "associative"; that is, $(AB)C = A(BC)$ (with numbers, $(3 \times 4) \times 5 = 3 \times (4 \times 5)$. But, unlike ordinary numbers, matrix multiplication is not "commutative". That is $AB \neq BA$ (but with numbers, $3 \times 4 = 4 \times 3$).

It is the associativity of matrix multiplication that allows us to build up a sequence of matrix operations representing elimination. Let us return once again to the matrix equation we started with:

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix}$$

We have seen, how, by multiplying both sides by the elimination matrix E (which we will now call E_{21}, for reasons that will be obvious), gives:

$$E_{21}(A\mathbf{x}) = (E_{21}A)\mathbf{x} = E_{21}\mathbf{b}$$

or:

$$A_1\mathbf{x} = E_{21}\mathbf{b}$$

where $A_1 = E_{21}A$. Without more elaboration, we now simply present the elimination matrices E_{31} and E_{32} that correspond to the remaining two elimination steps.

$$E_{31} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad E_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$$

The general rule for constructing an elimination matrix is this. If we are looking at n equations in m unknowns, and an elimination step involves multiplying equation j by a number q and subtracting it from equation i, then the elimination matrix E_{ij} is simply the $n \times m$ "identity matrix" I, with $a_{ij} = 0$ in I replaced by $-q$. For example, with 3 equations in 3 unknowns, and an elimination step that "multiplies equation 2 by 2 and subtracts from equation 3":

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad E_{32} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}$$

Each elimination step therefore results in a multiplication of both sides of $A\mathbf{x} = \mathbf{b}$ by the corresponding elimination matrix. In our example, the three elimination steps give:

$$E_{32}E_{31}E_{21}(A\mathbf{x}) = E_{32}E_{31}E_{21}\mathbf{b}$$

which, using the property of associativity of matrix multiplication is:

$$(E_{32}(E_{31}(E_{21}A)))\mathbf{x} = (E_{32}E_{31}E_{21})\mathbf{b}$$

Or:

$$U\mathbf{x} = (E_{32}E_{31}E_{21})\mathbf{b} = \mathbf{c} \tag{2.4}$$

where U is the upper triangular matrix $E_{32}A_2 = E_{32}(E_{31}A_1 = E_{32}(E_{31}(E_{21}A))$. Here:

$$U = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix} \quad \mathbf{c} = \begin{bmatrix} 2 \\ 6 \\ -10 \end{bmatrix} \tag{2.5}$$

Before we leave this section, there is one aspect of elimination that we have not yet considered. Let us look at the same equations as before, but in the following order:

$$\begin{aligned} 4y + z &= 2 \\ x + 2y + z &= 2 \\ 3x + 8y + z &= 12 \end{aligned}$$

The coefficient matrix A is then:

$$A = \begin{bmatrix} 0 & 4 & 1 \\ 1 & 2 & 1 \\ 3 & 8 & 1 \end{bmatrix} \tag{2.6}$$

Now clearly, no amount of elimination can get this matrix into an upper triangular form, since that would require a non-zero entry for a_{11}. Since there is no reason to keep the equations in this particular order, we can exchange their order until we reach the one we had in the previous section. Just as a single elimination step can be expressed as multiplication by an elimination matrix, exchange of a pair of equations can be expressed by multiplication by a *permutation* matrix.

The general rule for constructing a permutation matrix is this. If we are looking at m equations in n unknowns, and we want to exchange equations i and j, then the permutation matrix P_{ij} is simply the $m \times n$ "identity matrix" I, with rows i and j swapped:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad P_{12} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Multiplying a matrix A by P_{12} will swap rows 1 and 2 of A:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 4 & 1 \\ 1 & 2 & 1 \\ 3 & 8 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 4 & 1 \\ 3 & 8 & 1 \end{bmatrix}$$

What happens if, in spite of all exchanges, elimination still results in a 0 in any one of the pivot positions? Then we consider the process to have failed, and the equations do not have a solution. Assuming this does not happen, we will reach a point where the original equation $A\mathbf{x} = \mathbf{b}$ is transformed into $U\mathbf{x} = \mathbf{c}$ (as we did in Equation 2.4). The final step is that of *back-substitution*, in which variables are progressively assigned values using the right-hand side of this transformed equation (in Equation 2.3.1, $z = -2$, back-substituted to give $y = 1$, which finally yields $x = 2$).

2.4 Solution of Linear Equations by Matrix Inversion

So, it is possible to represent the steps leading to the solution of a set of linear equations by elimination entirely as a sequence of matrix multiplications. We now look at obtaining the solution by considering matrix "inversion". What we are trying to do is to really find a matrix analog for division with ordinary numbers. There, with an equation like $5x = 20$, we are able to get the answer immediately using division: $x = 20/5$. Can we not do the same with matrices? That is, given $A\mathbf{x} = \mathbf{b}$, can we not get $\mathbf{x} = \mathbf{b}/A$. Well, not quite. But we can get close: we find $\mathbf{x} = A^{-1}\mathbf{b}$, where A^{-1} is the matrix equivalent of $1/A$, and is called the *inverse* of the matrix.

2.4.1 Inverse Matrices

The starting point is just the same as with numbers. We know $a/a = aa^{-1} = 1$ for a non-zero number a. For matrices, we want to find A^{-1} such that $AA^{-1} = I$ where I is the identity matrix. Actually, with matrices, we can ask for inverses in two different ways: AA^{-1} and $A^{-1}A$, called for obvious reasons, right and left inverses of A (recall that since matrix multiplication does not necessarily commute, these could be different).

Let us start with $m \times n$ ("square") matrices. Our definition of an inverse is simply this: if there exists a matrix A_L^{-1} such that $A_L^{-1}A = I$, where I is the $N \times N$ identity matrix, then A_L^{-1} is called the left inverse of A. On the other hand, if there exists a matrix A_R^{-1} such that $AA_R^{-1} = I$, then A_R^{-1} is called the right inverse of A. Now, for square matrices, it is easy to see that the left and right inverses are the same:

$$A_L^{-1}(AA_R^{-1}) = (AA_L^{-1})A_R^{-1}$$

Or,

$$A_L^{-1} = A_R^{-1}$$

So, for square matrices at least, we can simply talk about "the inverse" A^{-1}. The question that now concerns us is: do all square matrices have an inverse? The short answer is "no". Here is a matrix that is not invertible:

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 6 \end{bmatrix} \tag{2.7}$$

We can see the conditions under which an inverse exists by referring back to the matrix equation that formed the basis of solution by elimination:

$$A\mathbf{x} = \mathbf{b}$$

Let us assume that A^{-1} exists. Then, the solution reached by elimination would simply be:

$$\mathbf{x} = A^{-1}\mathbf{b} \tag{2.8}$$

Therefore, if the inverse exists, then elimination must produce an upper triangular matrix with non-zero pivots. In fact, the condition works both ways—if elimination produces non-zero pivots then the inverse exists (you can see very quickly that elimination applied to the matrix A in Equation 2.4.1 would give give a row of 0s). Otherwise, the matrix is not invertible, or *singular*. Another way to look at this is that the matrix will be singular if its "determinant" is 0. We will look at what this means later (in Section 2.10), but it is related to the elimination producing non-zero pivots.

If the inverse exists, then the only solution to the matrix equation $A\mathbf{x} = \mathbf{b}$ is $\mathbf{x} = A^{-1}\mathbf{b}$. This gives another way to test for the singularity of a matrix: if there are solutions other than $\mathbf{x} = \mathbf{0}$ to $A\mathbf{x} = \mathbf{0}$. For example, with A in Equation 2.4.1, the vector $\mathbf{x} = [3, -1]$ is a solution to $A\mathbf{x} = \mathbf{0}$.

A final observation may be evident from the example in Equation 2.4.1. A matrix is singular if the columns (or rows) are not linearly independent.

Now let us consider a slight variant of the matrix A in Equation 2.4.1:

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 7 \end{bmatrix}$$

We believe that this matrix is invertible. How can we determine it's inverse? Let the inverse be

$$A^{-1} = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \tag{2.9}$$

The system of equations $AA^{-1} = I$ can be written as:

$$\begin{bmatrix} 1 & 3 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} a & c \\ b & d \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Again, recall the view of matrix multiplication in which each column on the right hand side is a linear combination of the columns of A:

$$\begin{bmatrix} 1 & 3 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 3 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

So, once we solve these two sets of linear equations, we can assemble A^{-1} from the values of a, b, c, and d. We are back, therefore, to solving linear systems of equations— the Gaussian elimination procedure for a single set of linear equations with a single column vector on the right-hand side has to be generalised. The process used is called the *Gauss-Jordan* procedure.

2.4.2 Gauss-Jordan Elimination

The Guass-Jordan elimination method addresses the problem of solving several linear systems $A\mathbf{x}_i = \mathbf{b}_i$ $(1 \leq i \leq N)$ at once, such that each linear system has the same coefficient matrix A but a different right hand side b_i.

From Section 2.3, we know that Gaussian elimination is nothing more than multiplication by elimination matrices, that transforms a coefficient matrix A into an upper-triangular matrix U:

$$U = E_{32}(E_{31}(E_{21}A)) = (E_{32}E_{31}E_{21})A$$

Here E_{ij} is an elimination matrix constructed as we described before (replacing the appropriate 0 in the identity matrix with a non-zero number). Of course, we might, in general, be required to perform row permutation operations and they will simply as appear as multiplication by permutation matrices. But, let us ignore this complication for the moment. Suppose now we applied further elimination steps until U was transformed into the identity matrix. This means multiplication by more matrices:

$$I = E_{13}(E_{12}(E_{23}(E_{32}(E_{31}(E_{21}A))))) = (E_{13}E_{12}E_{23}E_{32}E_{31}E_{21})A = BA \tag{2.10}$$

By definition $B = (E_{13}E_{12}E_{23}E_{32}E_{31}E_{21})$ must be A^{-1}. And this is what Gauss-Jordan does: it simply runs the elimination steps further until the upper-triangular matrix is converted into the identity matrix. So, A^{-1} can be computed by applying the same sequence of elimination steps to the identity matrix. A standard technique for carrying out the same elimination steps on two matrices A and B is to create an augmented matrix $[A\ B]$ and carry out the elimination on this augmented matrix. Gauss-Jordan can therefore be summarised in a single line: perform elimination steps on the augmented matrix $[A\ I]$ (representing the equation $AB = I$) to give the augmented matrix $[I\ A^{-1}]$ (representing the equation $IB = A^{-1}$). Or, in matrix multiplication terms: We illustrate the process with the example matrix we looked at earlier:

$$\left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 2 & 7 & 0 & 1 \end{array}\right] \overset{Row_2-2\times Row_1}{\Longrightarrow} \left[\begin{array}{cc|cc} 1 & 3 & 1 & 0 \\ 0 & 1 & -2 & 1 \end{array}\right] \overset{Row_1-3\times Row_2}{\Longrightarrow} \left[\begin{array}{cc|cc} 1 & 0 & 7 & -3 \\ 0 & 1 & -2 & 1 \end{array}\right]$$

One could verify that the inverse of A is given by

$$A^{-1} = \begin{bmatrix} 7 & -3 \\ -2 & 1 \end{bmatrix} \tag{2.11}$$

Gauss-Jordan therefore gives us a method to construct the inverse of a coefficient matrix A, and therefore directly solve $A\mathbf{x} = \mathbf{b}$ as $\mathbf{x} = A^{-1}\mathbf{b}$.

What if A is not a square matrix but rather a rectangular matrix of size $m \times n$, such that $m \neq n$. Does there exist a notion of A^{-1}? The answer depends on the rank of A.

- If A is full row rank and $n > m$, then AA^T is a full rank $m \times m$ matrix and therefore $(AA^T){-}1$ exists. The matrix $A^T(AA^T){-}1$ yields the identity matrix when multiplied to A on its right, *i.e.*, $AA^T(AA^T){-}1 = I$ and is called the right inverse of A. When the right inverse of A is multiplied on its left, we get the projection matrix $A^T(AA^T)^{-1}A$, which projects matrices onto the row space of A.

- If A is full column rank and $m > n$, then A^TA is a full rank $n \times n$ matrix and therefore $(A^TA){-}1$ exists. The matrix $(A^TA){-}1A^T$ yields the identity matrix when multiplied to A on its left, *i.e.*, $(AA^T){-}1A^TA = I$ and is called the left inverse of A. When the left inverse of A is multiplied on its right, we get the projection matrix $A(A^TA)^{-1}A^T$, which projects matrices onto the column space of A.

What if A is neither full row rank nor full column rank? In Section 2.13, we define the pseudoinverse of any $m \times n$ matrix, without any restrictions on its rank.

2.5 Solution of Linear Equations using Vector Spaces

We now turn to the third approach for solving linear equations. This is, in some sense, the most abstract, and involves the idea a vector spaces. A vector space is a collection of vectors that, informally speaking, may be multiplied by a number and added. More formally, a vector space is a set of vectors on which two operations are defined: vector addition and scalar multiplication. Additionally, these two operations satisfy certain natural conditions which we will elaborate shortly. A well-known example is the vector space $\Re^2$. This consists of all 2 dimensional column vectors with real-valued components (this is also just the entire xy plane). Similarly, the space $\Re^n$ comprises all n dimensional column vectors with real-valued components.

More generally, if a set of vectors $\mathcal{V}$ is to qualify as a "vector space" then two vector operations—addition and scalar multiplication—have to be defined, and they have to result in vectors within the set $\mathcal{V}$. The set, or space $\mathcal{V}$ is then said to be "closed" under the operations of addition and scalar multiplication. Now, given vectors $\mathbf{u}$ and $\mathbf{v}$ in a vector space, all scalar multiples of vectors $a\mathbf{u}$ and $b\mathbf{v}$ are in the space, as is their sum $a\mathbf{u} + b\mathbf{v}$. That is, all linear combinations of elements in the space are also elements of the space ((V) is closed under linear combination). If a subset (V_S) of any such space is itself a vector space (that is, (V_S) is also closed under linear combination) then (V_S) is called a subspace of (V). All this may seem a bit abstract, and some examples may help:

1. The set of vectors $\Re^2$ consisting of all two-dimensional vectors with real-valued components is a vector space. Adding any two vectors in the set gives a vector that is also in the set. Scalar multiplication of any vector in the set is a vector also in $\Re^2$ (you may find these easy to see by visualising the vectors in the xy plane).

2. The set of vectors $(\Re^2)^+$ consisting of all two-dimensional vectors in the positive quadrant is *not* a vector space. Adding a pair of vectors in $(\Re^2)+$ results in a vector in $(\Re^2)^+$). But, unfortunately, multiplying by a scalar may not. For example, every vector $-3\mathbf{v}$ ($\mathbf{v} \in (\Re^2)^+$) does not belong to $(\Re^2)^+$.

3. The subset $\Re^3_S$ of $\Re^3$ consisting of vectors of any length through the origin $(0, 0, 0)$ is a subspace of $\Re^3$. Adding vectors in $\Re^3_S$ clearly results in an element of the set, as does multiplication by a scalar. It is important that the origin $(0, 0, 0)$ is included: otherwise, multiplication of a vector by 0 would result in a vector not in the subset.

2.5.1 Vector Spaces and Matrices

Our condition on vector spaces has nothing really to do with vectors: all we need is that a pair of operations, addition and scalar multiplication be defined on a set of elements. So, we are now ready to go a further step, and drop the restriction that vector spaces consist only of vectors. We can, for example, talk of a "vector" space $\mathcal{M}$ consisting of all 2×2 matrices. It is easy to check that this is indeed closed under (matrix) addition and scalar multiplication (we have not come across this before: it is simply the multiplication of every element of the matrix by the number comprising the scalar). Just as with vectors, a subspace of $\mathcal{M}$ is then some subset that is also a vector space.

Vector spaces of matrices provide a novel way of looking at the solution of $A\mathbf{x} = \mathbf{b}$. Recall that $A\mathbf{x}$ is simply a linear combination of the columns of the matrix A. All possible linear combinations of the columns produce a set of all possible column vectors (in effect, all possible $\mathbf{b}$'s). This set is called the *column space* of A, or $C(A)$. Given $\mathbf{b}$, therefore, when we ask: is there a solution to $A\mathbf{x} = \mathbf{b}$, we are really asking if the particular $\mathbf{b}$ we are given is in the column space of A. An example may help. Consider the matrix A:

$$A = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 4 \\ 4 & 1 & 5 \end{bmatrix}$$

The column space $C(A)$ is a subspace of $\Re^4$ (are you sure you understand why this is so?). We can ask an number of questions now. What is in this subspace? Obviously, each column of A is in $C(A)$. Additionally, $C(A)$ contains all linear combinations of the columns of A. Is $C(A)$ the entire 4$-$dimensional space $\Re^4$? If not, how much smaller is $C(A)$ compared to $\Re^4$?

Equivalently, we can pose this problem as follows. Consider the linear system $A\mathbf{x} = \mathbf{b}$. For which right hand sides $\mathbf{b}$ does a solution $\mathbf{x}$ always exist? A solution $\mathbf{x}$ definitely does not exist for every right hand side $\mathbf{b}$, since there are 4 equations in 3 unknowns. Let us analyse this further by writing down the system of equations

$$A\mathbf{x} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 4 \\ 4 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.12}$$

Our first point is that there are many vectors $\mathbf{b}$ which cannot be expressed as the linear combination of the columns of A. That leads to the question, *which right hand side* $\mathbf{b}$ *allows the equation to be solved.* One value of $\mathbf{b}$ for which the equation can be solved is the zero vector, for which the corresponding solution is $\mathbf{x} = \mathbf{0}$. Three other trivial values of $\mathbf{b}$ are the values corresponding to every column of A. In particular, we can solve $A\mathbf{x} = \mathbf{b}$ whenever b is in the column space $C(A)$. When $\mathbf{b}$ is a combination of columns of A, the combination tells us what exactly $\mathbf{x}$ must be.

Do all the columns of A contribute something 'new' to the space $C(A)^1$? In other words, can we get the same space $C(A)$ using less than three columns of A? In this particular example, the third column of A is a linear combination of the first two columns of A. $C(A)$ is therefor a $2-$dimensional subspace of $\Re^4$. In general, if A is an $m \times n$ matrix, $C(A)$ is a subspace of $\Re^m$.

2.5.2 Null Space

The null space of a matrix A, referred to as $N(A)$, is the space of all solutions to the equation $A\mathbf{x} = 0$. The null space of an $m \times n$ matrix A is a subspace of $\Re^n$.

Cosinder the example matrix A discussed in the previous section. Its null space is a subspace of $\Re^3$. We will try to figure out the null space of the matrix A by observing the following system of equations:

$$A\mathbf{x} = \begin{bmatrix} 1 & 1 & 2 \\ 2 & 1 & 3 \\ 3 & 1 & 4 \\ 4 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

One obvious solution to the system is the zero vector. The null space will always contain the zero vector. Making use of the observation that the columns of A are linearly dependent, we find a second solution to the system as:

$$\mathbf{x}^* = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \tag{2.14}$$

Thus, $\mathbf{x}^*$ is in the null space $N(A)$. Every multiple of $\mathbf{x}^*$ is also in the null space. Each element of the null space $N(A)$ is of the form

$$c.\mathbf{x}^* = \begin{bmatrix} c \\ c \\ -c \end{bmatrix} \tag{2.15}$$

[1]In subsequent sections, we will refer to these columns as *pivot* columns.

where $c \in \Re$. Thus, the null space $N(A)$ is the line passing through the zero vector $[0\ 0\ 0]$ and $[1\ 1\ -1]$.

Do solutions to $Ax = 0$ always yield a vector space? The answer is yes and this can be proved by observing that if $A\mathbf{v} = 0$ and $A\mathbf{w} = 0$, then $A(\mathbf{v}+\mathbf{w}) = 0$ and $A(p\mathbf{v}) = 0$ where $p \in \Re$. In general, there are two equivalent ways of specifying a subspace.

1. The first way is to specify a bunch of vectors whose linear combinations will yield the subspace.
2. The second way is to specify a system of equations of the form $A\mathbf{x} = \mathbf{0}$ and any vector $\mathbf{x}$ the satisfies the system is an element of the subspace.

What about the set of all solutions to the equation $A\mathbf{x} = \mathbf{b}$ - do elements of this set form a space? The answer is a *no*. An easy way to see this is that the zero vector is not a solution to this system (unless $\mathbf{b}$ is the zero vector) and hence the solutions cannot form a space.

2.6 Elimination for Computing the Null Space (Ax = 0)

In the last section we defined the null space of a matrix A. In this section, we will turn the definition into an algorithm using the elimination technique discussed in Section 2.3. We will take as an example, the following rectangular matrix A

$$A = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 8 & 10 \end{bmatrix} \tag{2.16}$$

2.6.1 Pivot Variables and Row Echelon Form

We note rightaway that column 2 of A is a multiple of column 1 - it is in the same direction as column 1 and is therefore not indepedent. We expect to discover that in the elimination process for computing the null space $N(A)$. In terms of rows, we observe that the third row is the sum of the first two rows. Thus, the rows are also not independent - and again we hope to discover that in the elimination process.

In essence, what elimination does is change the matrix A and consequently its column space, while leaving the null space of A intact. We first choose the element in position $[1, 1]$ as the pivot element and perform the steps $(2, 1)$ and $(3, 1)$ (recall the definition of a step from Section 2.3) to get the transformed matrix A_1.

$$A_1 = \begin{bmatrix} [1] & 2 & 2 & 2 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 2 & 4 \end{bmatrix} \tag{2.17}$$

We have got the first column in the desirable form. So next, we try to use the element in position $[2, 2]$ as the pivot element. But unfortunately it is a 0. We look below it position $[3, 2]$ hoping for a non-zero element so that we can do a row exachange. But there is a zero below as well! That tells us that second column is dependent on the first column.

Since we have nothing to do with the second column, we move to the thrid column and choose the entry $[2, 3]$ as the pivot. We perform the next elimination step $(3, 2)$, and obtain a third row of zeros. We will denote the resultant matrix by U. Note that the pivots are marked in boxes.

$$U = \begin{bmatrix} [1] & 2 & 2 & 2 \\ 0 & 0 & [2] & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.18}$$

The matrix U is in the row echelon form. A matrix is said to be in row echelon form if it satisfies the following requirements:

1. All nonzero rows are above any rows of all zeroes.
2. The leading coefficient of a row is always strictly to the right of the leading coefficient of the row above it.

While reducing from the matrix A to U, we used only two pivots. In fact, we have already discovered the most important number about the matrix A. The number of pivots is 2 - which is also the *rank* of the matrix.

Fact: *The rank of a matrix is the number of pivots used while reducing it to the row echelon form using elimination.*

We can now solve $U\mathbf{x} = \mathbf{0}$, which has the same solution as $A\mathbf{x} = \mathbf{0}$ (since the elimination steps on zero vector always yield a zero vector). Thus, the null space of U is the same as that of A. How do we describe these solutions? We will first write down the equations corresponding to $U\mathbf{x} = \mathbf{0}$.

$$x_1 + 2x_2 + 2x_3 + 2x_4 = 0$$
$$2x_3 + 4x_4 = 0$$

We will descrie the solution by first separating out the two columns containing the pivots, referred to as *pivot columns* and the remaining columns, referred to as *free columns*. Variables corresponding to the free columns are called *free variables*, since they can be assigned any value. Variables corresponding to the pivot columns are called *pivot variables*, and their values can be determined based on the values assigned to the free variables. In our example, variables x_2 and x_4 are free variables while x_1 and x_3 are the pivot variables.

Let us say we use the following assignment of values to free variables: $x_2 = 1$, $x_4 = 0$. Then, by back substition, we get the following values: $x_1 = -2$ and $x_3 = 0$. Thus, the following vector $\mathbf{x}'$ is a solution to the system $U\mathbf{x} = \mathbf{0}$ (and consequently the solution to $A\mathbf{x} = \mathbf{0}$) and therefore lies in $N(A)$.

$$\mathbf{x}' = \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.19}$$

This solution reiterates our first observation, *viz.*, that column 2 is a multiple of column 1.

We will find some more vectors in the null space. Any multiple $\mathbf{c}.\mathbf{x}', c\Re$ is also in $N(A)$. Note that $\mathbf{c}.\mathbf{x}'$ is a line. Are these the only vectors in $N(A)$? Actually, no – we obtained this set of vectors by assigning only one set of values to the free variables x_2 and x_4. We assign another set of values $x_2 = 0$, $x_4 = 1$, and obtain the values of x_1 and x_3 by back-substitution to get another vector $\mathbf{x}''$ in $N(A)$.

$$\mathbf{x}'' = \begin{bmatrix} 2 \\ 0 \\ -2 \\ 1 \end{bmatrix} \tag{2.20}$$

Now we are in a position to specify all vectors in $N(A)$. The null space will contain all linear combinations of the two special solutions $\mathbf{x}'$ and $\mathbf{x}''$. Every vector in $N(A)$ therefore has the form given in (2.21):

$$ax' + bx'', a \in \Re, b \in \Re \tag{2.21}$$

What is the number of special (linearly independent) solutions for $A\mathbf{x} = \mathbf{0}$ if A is an $m \times n$ matrix? As we saw earlier, the rank r of a matrix equals the number of pivots. The number of free variables is given by

$$no.\ of\ free\ variables\ =\ n -\ r$$

The number of special solutions is exactly equal to the number of free variables. In the above example, we had $n = 4, r = 2$ and therefore number of free variables was 2. The steps for characterizing the null space of a matrix A can be summarized as follows:

1. Reduce A to the row echelon form.
2. If r is the number of pivots, find the $k = n - r$ free variables.
3. Make k different assignments to the free variables. For each assignment, use backsubstitution (using the row echelon form) to find the values of the pivot variables. Each assignemt to the free variables yields a vector in the null space.

2.6.2 Reduced Row Echelon Form

We will take a second look at the matrix U that we obtained by elimination.

$$U = \begin{bmatrix} [1] & 2 & 2 & 2 \\ 0 & 0 & [2] & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.22}$$

The last row of U is composed of zeroes. This is because row 3 of A was a linear combination of rows 1 and 2 and this fact was discovered by elimination. How can we clean U up further? We can do elimination upwards to get zeros above as well as below the pivots. Elimination step $(2, 1)$ on U yields the matrix U'.

$$U' = \begin{bmatrix} [1] & 2 & 0 & -2 \\ 0 & 0 & [2] & 4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.23}$$

Further, we will make all pivot elements equal to 1 by dividing the corresponding row by the pivot element. This yields the matrix R.

$$R = \begin{bmatrix} [1] & 2 & 0 & -2 \\ 0 & 0 & [1] & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.24}$$

The matrix R has zeroes above and below each pivot. This matrix is called the reduced row echelon form (rref) of A. Matlab has the function *rref(A)* that returns the reduced row echelon form of A. The system of equations $R\mathbf{x} = 0$ is given as

$$x_1 + 2x_2 - 2x_4 = 0$$

$$x_3 + 2x_4 = 0$$

The solution to this system is the same the solution to the original system of equations $Ax = 0$. By simple back-substitution, the vector x can be expressed as:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ 1 & 0 \\ 0 & -2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_2 \\ x_4 \end{bmatrix} \tag{2.25}$$

Note that the specification of the null space in equation 2.25 is the same as that in equation 2.21.

Let us suppose that we have already got a matrix A in the reduced row echelon form (rref) R. Further, let us pretend that the pivot columns I come before the free columns F. The matrix R can be written as

$$R = \begin{bmatrix} I & F \\ 0 & 0 \end{bmatrix} \tag{2.26}$$

This block matrix is a very typical rref. We can easily do all the special solutions at once. We will create a *null basis* N whose columns are the special solutions; *i.e.*, $RN = 0$. The following N satisfies this system:

$$N = \begin{bmatrix} -F \\ I \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ 0 & -2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.27}$$

In fact there is a Matlab command $null(A)$ that returns the null basis of A. It first computes the rref of A and then composes N using the free columns of A as well as the identity matrix of size equal to the rank of A.

Next, we will llustrate the same sequence of steps on the transpose matrix A^t to obtain its row echelon form U and observe the pivots, rank and null space. The solution to $A^t\mathbf{x} = \mathbf{0}$ is a column vector of size 3. Notice that the rank of the transpose is again 2 and is the same as the number of pivot columns. There is a single free column corresponding to the free variable x_3.

$$\begin{bmatrix} [1] & 2 & 3 \\ 2 & 4 & 6 \\ 2 & 6 & 8 \\ 2 & 8 & 10 \end{bmatrix} \overset{E_{2,1},E3,1,E4,1}{\Longrightarrow} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 2 & 2 \\ 0 & 4 & 4 \end{bmatrix} \overset{P_{2,3}}{\Longrightarrow} \begin{bmatrix} 1 & 2 & 3 \\ 0 & [2] & 2 \\ 0 & 0 & 0 \\ 0 & 4 & 4 \end{bmatrix} \overset{E_{4,2}}{\Longrightarrow} \begin{bmatrix} [1] & 2 & 3 \\ 0 & [2] & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.28}$$

Suppose we make the following assignment to the free variable $x_3 = -c$. Then the solution is given by

$$\begin{bmatrix} -c \\ -c \\ c \end{bmatrix} = c \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \tag{2.29}$$

Thus, the null space of A^t is a line. Taking the elimination steps forward, we can get the reduced row echelon form (as a block matrix) R for matrix A^t.

$$\begin{bmatrix} [1] & 2 & 3 \\ 0 & [2] & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \overset{E_{1,2}}{\Longrightarrow} \begin{bmatrix} [1] & 0 & 1 \\ 0 & [2] & 2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \overset{(\frac{Row_2}{2})}{\Longrightarrow} \begin{bmatrix} [1] & 0 & 1 \\ 0 & [1] & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \overset{of\ the\ form}{\Longleftrightarrow} \begin{bmatrix} I & F \\ 0 & 0 \end{bmatrix} \tag{2.30}$$

The null basis N is

$$N = \begin{bmatrix} -F \\ I \end{bmatrix} = \begin{bmatrix} -F \\ I \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix} \tag{2.31}$$

2.6.3 Solving $A\mathbf{x} = \mathbf{b}$

In this sub-section we will illustrate how to completely solve the system $A\mathbf{x} = \mathbf{b}$, if there is a solution. If there is no solution, we will need to identify this fact and if there is some solution, we need to identify how many solutions it has. Our running example will be a system of equations with the same coefficient matrix A that was considered in the previous section (2.6).

$$A\mathbf{x} = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 2 & 4 & 6 & 8 \\ 3 & 6 & 8 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.32}$$

The third row is the sum of rows 1 and 2. In other words, if we add the left hand sides, we get the third left hand sides. So we can predict right away what elimination will discover about the right hand side. What is the condition that b_1, b_2 and b_3 satisfy so that there is a solution? Since the sum of the first two left hand sides equals the third left hand side, we require that $b_1 + b_2 = b_3$.

Let us see how elimination discovers this fact. If some combination on the left hand side gives zeros, the same combination on the right hand side should give zeros. Tacking on the vector of **b**'s as another column to the matrix A, we get the augmented matrix $[A\ \mathbf{b}]$. Applying the elimination steps $E_{2,1}$ and $E_{3,1}$ to the augmented matrix, followed by the elimination step $E_{3,2}$, we get:

$$[A\ \mathbf{b}] = \begin{bmatrix} 1 & 2 & 2 & 2 & b_1 \\ 2 & 4 & 6 & 8 & b_2 \\ 3 & 6 & 8 & 10 & b_3 \end{bmatrix} \overset{E_{2,1},E3,1}{\Longrightarrow} \begin{bmatrix} [1] & 2 & 2 & 2 & b_1 \\ 0 & 0 & [2] & 4 & b_2 - 2b_1 \\ 0 & 0 & 2 & 4 & b_3 - 3b_1 \end{bmatrix}$$
$$\overset{E3,2}{\Longrightarrow} \begin{bmatrix} [1] & 2 & 2 & 2 & b_1 \\ 0 & 0 & [2] & 4 & b_2 - 2b_1 \\ 0 & 0 & 0 & 0 & b_3 - b_1 - b_2 \end{bmatrix} \tag{2.33}$$

The condition for solvability is therefore $b_3 - b_1 - b_2 = 0$. Thus, the system of equations will have a solution for $\mathbf{b} = [5\ 1\ 6]^T$.

We will now discuss the solvability conditions on the right hand side of a system of equations to ensure that the system of equations $A\mathbf{x} = \mathbf{b}$ is solvable. We will provide a definition in terms of the column space.

The system of equations $A\mathbf{x} = \mathbf{b}$ is solvable when **b** *is in the column space $C(A)$.*

Another way of describing solvability is:

The system of equations $A\mathbf{x} = \mathbf{b}$ is solvable if a combination of the rows of A produces a zero row, the requirement on **b** *is that the same combination of the components of* **b** *has to yield zero.*

It is not immediately apparent that the two systems of equations are equivalent. We will come back to discuss this in a later part of the course. We will proceed to the case when the system of equations does have a solution.

Assuming that the systems of equations $A\mathbf{x} = \mathbf{b}$ is solvable, what is the algorithm (or sequence of steps) to find the complete solution? We will start by finding one particular solution.

1. $\mathbf{x}_{particular}$[2]: Set all free variables (corresponding to columns with no pivots) to 0. In the example above, we should set $x_2 = 0$ and $x_4 = 0$.

2. Solve $A\mathbf{x} = \mathbf{b}$ for pivot variables.

This leaves us with the equations

$$x_1 + 2x_3 = b_1 2x_3 = b_2 - 2b_1$$

Adopting the normal back substitution method, we get

$$x_3 = \frac{b_2 - 2b_1}{2} x_1 = b_2 + 3b_1 \tag{2.34}$$

Thus the particular solution is

$$\mathbf{x}_{particular} = \begin{bmatrix} b_2 + 3b_1 \\ 0 \\ \frac{b_2 - 2b_1}{2} \\ 0 \end{bmatrix}$$

For example, if we choose $\mathbf{b} = [5\ 1\ 6]^T$, we get

$$\mathbf{x}_{particular} = \begin{bmatrix} -2 \\ 0 \\ \frac{3}{2} \\ 0 \end{bmatrix}$$

The sequence of steps is (a) check for solvability conditons (b) substitute some values for the free variables and obtain values for pivot variables. How do we find the complete solution to $A\mathbf{x} = \mathbf{b}$? It is easy to see that any vector $\mathbf{x}_{nullspace}$ in the null space of the matrix A can be added to $\mathbf{x}_{particular}$ and the resultant vector will still remain a solution. Thus, a general solution to the system $A\mathbf{x} = \mathbf{b}$ is

$$\mathbf{x}_{complete} = \mathbf{x}_{particular} + \mathbf{x}_{nullspace} \tag{2.35}$$

Let us write out the complete solution to this example (recall the null space for this matrix from Equation 2.25).

[2]Since there are many solutions, one could have one's own way of finding one solution.

$$x_{complete} = \begin{bmatrix} -2 \\ 0 \\ \frac{3}{2} \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 2 \\ 1 & 0 \\ 0 & -2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \tag{2.36}$$

This pattern shows up in all of mathematics, whenever we have linear equations. The reason for this is that

$$A\mathbf{x}_{complete} = A(\mathbf{x}_{particular} + \mathbf{x}_{nullspace}) = \mathbf{b} + \mathbf{0} = \mathbf{b}$$

In words, this means that if we have one solution, we can add on anything in the null space. This gives us the 'complete' solution. Note that while the null vector can be scaled arbitrarily, the same does not hold for the *particular* solution.

Let us say we want to plot all solutions to this equation. The plot should be in $\Re^4$ because there are 4 unknowns. Does the set of solutions to $A\mathbf{x} = \mathbf{b}$ form a subspace? No, because the space of solutions to this system is not closed under the scaling operation. The null space is a 2-dimensional[3] subspace inside $\Re^4$. The set of solutions to $Ax = b$ does not however pass through the origin, because it must pass through $\mathbf{x}_{particular}$ and then onward. It is like a sub-space shifted away from the origin!

In summary, the algorithm is to go through elimination, find a particular solution and then a special solution. We will now visualize the bigger picture by answering some questions. Consider an $m \times n$ matrix A of rank r.

Q1. What is the relationship between m and r? We know certainly that $r \leq m$ and $r \leq n$. This because, each row as well as column can contain only one pivot and therefore the number of pivots should be less than the number of rows as also less than the number of columns.

Q2. What happens when the rank r is as big as it can be? There are two possibilities here, depending on what the numbers m and n are.

Q3. In the case that A is full column rank, i.e., $r = n$, what can we infer about the null space and the complete solution? Full column rank implies that there is a pivot in every column, that is, there are n pivots. As a result,there are no free variables. The implication is that the null space will only have the $\mathbf{0}$ vector. Therefore, the complete solution is just $\mathbf{x}_{particular}$; there is just one solution, if there is one at all. Thus, the number of solutions is either 0 or 1. There are many applications in reality where the columns are independent and have nothing to look for in the null space, leading to just a particular solution.

We will illustrate by squeezing in an example.

[3]The dimension of the subspace corresponds to the number constants you can choose.

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 1 \\ 6 & 1 \\ 5 & 1 \end{bmatrix} \tag{2.37}$$

The rank of this matrix is 2; elimination will yield exactly 2 pivots. Carrying out the elimination process to the end, we can get the following reduced row echelon form for this matrix:

$$A_{rref} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -17 \\ 0 & -14 \end{bmatrix} \tag{2.38}$$

The first two rows are not independent, but the other rows are combinations of the first two rows. It is a case of full column rank. $A\mathbf{x} = \mathbf{b}$ is a system of four equations in two unknowns. If the right hand side is not consistent with the 4 equations, we will get zero solutions. The right hand side $b = [4 \;\; 3 \;\; 7 \;\; 6]^T$ is consistent with the equations and yields one solution. Similarly, the right hand side b which is the sum of the two independent columns of A also gives one unique solution $\mathbf{x} = [1 \; 1]^T$. We will maintain the natural symmetry of this discussion by next looking at *full row rank*.

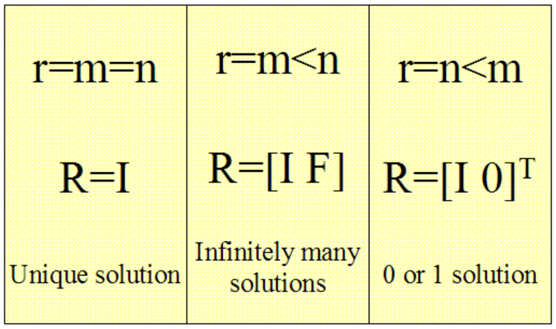
Q4. In the case that A is full row rank, i.e., $r = m$, what can we infer about the null space and the complete solution? Elimination will lead to m pivots; every row will have a pivot. What is the implication on solvability, *i.e.*, for which right hand sides will we have a solution to $A\mathbf{x} = \mathbf{b}$? Since we do not have any zero rows, we can solve the system for every right hand side $\mathbf{b}$. This resolves the question about the existence of a solution. How many free variables will we have? Since $n \geq r$, we will be left with $n - r = n - m$ free variables.

An easy way to obtain an example here (matrix B) is to transpose the above full column rank example matrix A.

$$B = A^T = \begin{bmatrix} 1 & 2 & 6 & 5 \\ 3 & 1 & 1 & 1 \end{bmatrix} \tag{2.39}$$

Elimination yields the following row reduced echelon form with two pivots:

$$\begin{bmatrix} 1 & 0 & 4 & 3 \\ 0 & 1 & -11 & -8 \end{bmatrix} \tag{2.40}$$



r=m=n	r=m<n	r=n<m
R=I	R=[I F]	R=[I 0]^T
Unique solution	Infinitely many solutions	0 or 1 solution

Figure 2.3: Summary of the properties of the solutions to the system of equations $Ax = b$.

The number of free variables is 2.

Q5. In the case that A is full rank, i.e., $r = m = n$, what can we infer about the null space and the complete solution?

This is the most important case of all. We will illustrate with an example.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 1 \end{bmatrix} \tag{2.41}$$

The reduced row echelon form for this matrix is

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.42}$$

The matrix is invertible; invertible matrices come out naturally in the rref which is the identity matrix. Which are the satisfiable right hand sides $\mathbf{b}$ for the system $A\mathbf{x} = \mathbf{b}$? Since there are no zero rows, there are no constraints on the right hand side. What is the null space of A? It is the zero vector only. Since the rank is also m, the only solution is the *particular solution*, and is therefore a unique solution.

Figure 2.3 summarizes the properties of the solutions to the system of equations $Ax = b$ for different inequality constraints between m, n and r. The rank summarizes the possible solutions, except the exact entries in the solutions.

2.7 Independence, Basis, and Dimension

In this section, we will develop the ideas of linear independence of vectors, the space vectors span, basis for vector spaces and finally the dimension of vector spaces. We will assign clear meanings to these terms.

To set the appropriate background, we will begin with a highly important fact which was mentioned earlier. Let us say we have the system $A\mathbf{x} = 0$, where A is an $m \times n$ matrix and $m < n$. That is, we have more unknowns than equations. The conclusion is that there there are some non-zero vectors in the null space of A. If we perform elimination on A, we will get some pivots and some free columns that do not have pivots because there will be $n - m$ free variables. We can assign non-zero values to the free variables and automatically obtain values for the pivot variables. We will resume from this point.

2.7.1 Independence

Independence *Vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are independent if no linear combination gives the zero vector, except the zero combination. That is, $\forall c_1, c_2, \ldots, c_n \in \Re$, such that not all of the c_i's are simultaneously 0, $\sum_i^n c_i\mathbf{x}_i \neq \mathbf{0}$.*

For example, in a two dimensional space, a vector $\mathbf{x}$ and twice the vector $2\mathbf{x}$ are dependent, because $(-2) \times \mathbf{x} + (1) \times 2\mathbf{x} = \mathbf{0}$. As another example, suppose we have the vectors $\mathbf{v}_1$ and a zero vector vector $\mathbf{v}_2$, they are dependent because $(0) \times \mathbf{v}_1 + (100) \times \mathbf{v}_2 = \mathbf{0}$.

On the other hand, two non-zero vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ in a two dimensional space that make an angle $0 < \theta < \frac{\pi}{2}$ with each other will be independent. If we however add a third vector $\mathbf{v}_3$ from the two dimensional space to the set, the three vectors will now be dependent. How do we determine the truth of the above two statements? We could do this as follows. We construct a matrix A with the vectors as three columns $A = [v_1\ v_2\ v_3]$. This matrix is a 2×3 matrix. Does there exist a non-zero solution to the following system?

$$Ac = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.43}$$

It can be easily proved that a non-zero vector $[c_1\ c_2\ c_3]^T$ exists. We will restate the definition for independence in terms of the columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of a matrix A.

Independence *The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of a matrix A are independent if the null-space of A is the zero vector. The columns of A are dependent only if $A\mathbf{c} = 0$ for some $\mathbf{c} \neq \mathbf{0}$.*

In other words, the rank of the matrix A, whose columns are independent is the number of columns n. And in the reduced echelon form, all columns will be pivot columns with no free variables. If the columns are dependent, the rank of A will be less than n, and there will be free variables.

What does it mean for a bunch of vectors to span a space? Recall that we could take all combinations of the columns of a matrix to yield the column space. This column space is the space spanned by the columns of the matrix.

Space spanned by vectors: *Vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ span a space means that the space consists of all linear combinations of the vectors. Thus, the space spanned by the columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of a matrix A, is the column space of A.*

2.7.2 Basis and Dimension

The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$, need not be independent in order to span a space. We are specifically interested in a set of vectors that span a space and are at the same time linearly independent. This set of vectors is in some sense, the right number of vectors; even without a single vector from this set, the space cannot be defined. This set is called the *basis*.

Basis for a space: *The basis for a space is a set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ with two properties, viz., (1) The vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are independent and (2) These vectors span the space.*

The definition of basis is hinged on the preceeding two definitions - the basis is the set of vectors that is necessary and sufficient for spanning the space. As an example, one basis for the four dimensional space $\Re^4$ is:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.44}$$

It is easy to verify that the above vectors are independent; if a combination of the vectors using the scalars in $[c_1, c_2, c_3, c_4]$ should yield the zero vector, we must have $c_1 = c_2 = c_3 = c_4 = 0$. Another way of proving this is by making the four vectors the columns of a matrix. The resultant matrix will be an identity matrix. The null space of an identity matrix is the zero vector. The above basis is often called the *standard basis* for $\Re^4$.

This is not the only basis of $\Re^4$. Consider the following three vectors

$$\begin{bmatrix} 2 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \tag{2.45}$$

These vectors are certainly independent. But they do not span $\Re^4$. This can be proved by showing that the following vector in $\Re^4$ cannot be expressed as a linear combination of these vectors.

$$\begin{bmatrix} 0 \\ 2 \\ 0 \\ 0 \end{bmatrix} \tag{2.46}$$

In fact, if this vector is added to the set of three vectors in (2.45), together, they define another basis for $\Re^4$. And this could be proved by introducing them as columns of a matrix A, subject A to row reduction and check if there are any free variables (or equivalently, whether all columns are pivot columns). If there are no free variables, we can conclude that the vectors form a basis for $\Re^4$. This is also equivalent to the statement that *if the matrix A is invertible, its columns form a basis for its column space.* This statement can be generalized to $\Re^n$: *if an $n \times n$ matrix A is invertible, its coulumns for a basis for $\Re^n$.*

While there can be many bases for a space, a commonality between all the bases is that they have exactly the same number of vectors. This unique size of the basis is called the dimension of the space.

Dimension: *The number of vectors in any basis of a vector space is called the dimension of the space.*

Do the vectors in (2.45), form a basis for any space at all? The vectors are independent and therefore span the space of all linear combinations of the three vectors. The space spanned by these vectors is a hyperplane in $\Re^4$. Let A be any matrix. By definition, the columns of A span the column space $C(A)$ of A. If there exists a $\mathbf{c} \neq \mathbf{0}$ such that, $A\mathbf{c} = 0$, then the columns of A are not linearly independent. For example, the columns of the matrix A given below are not linearly independent.

$$A = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 2 & 3 & 5 & 2 \\ 3 & 4 & 7 & 3 \end{bmatrix} \tag{2.47}$$

A choice of $\mathbf{c} = [-1\ 0\ 0\ 1]^T$ gives $A\mathbf{c} = 0$. Thus, the columns of A do not form a basis for its columns space. What is a basis for $C(A)$? A most natural choice is the first two columns of A; the thid column is the sum of the first and second columns, while the fourth column is the same as the first column. Also, column elimination[4] on A yields pivots on the first two columns. Thus, a basis for $C(A)$ is

[4]Column elimination operations are very similar to row elimination operations.

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \tag{2.48}$$

Another basis for $C(A)$ consists of the first and third columns. We note that the dimension of $C(A)$ is 2. We also note that the rank of A is the number of its pivots columns, which is exactly the dimension of $C(A)$. This gives us a nice theorem.

Theorem 17 *The rank of a matrix is the same as the dimension of its column space. That is, $rank(A) = dimension\,(C(A))$.*

What about the dimension of the null space? We already saw that $\mathbf{c} = [-1\ 0\ 0\ 1]^T$ is in the null space. Another element of the null space is $\mathbf{c}' = [1\ 1\ -1\ 0]^T$. These vectors in the null space specify combinations of the columns that yield zeroes. The two vectors $\mathbf{c}$ and $\mathbf{c}'$ are obviously independent. Do these two vectors span the entire null space? The dimension of the null space is the same as the number of free variables, which happens to be $4 - 2 = 2$ in this example. Thus the two vectors c and c' must indeed span the null space. In fact, it can be proved that the dimension of the null space of an $m \times n$ matrix A is $n - rank(A)$.

The space spanned by the rows of a matrix is called the *row space*. We can also define the row space of a matrix A as the column space of its transpose A^T. Thus the row space of A can be specified as $C(A^T)$. The null space of A, $N(A)$ is often called the *right null space* of A, while the null space of A^T, $N(A^T)$ is often referred to as its *left null space*. How do we visualize these four spaces? $N(A)$ and $C(A^T)$ of an $m \times n$ matrix A are in $\Re^n$, while $C(A)$ and $N(A^T)$ are in $\Re^m$. How can we construct bases for each of the four subspaces? We note that dimensions of $C(A)$ and the rank of $C(A^T)$ should be the same, since row rank of a matrix is its column rank. The bases of $C(A)$ can be obtained as the set of the pivot columns.

Let r be the rank of A. Recall that the null space is constructed by linear combinations of the special solutions of the null space (2.5.2) and there is one special solution for each assignment of the free variables. In fact, the number of special solutions exactly equals the number of free variables, which is $n - r$. Thus, the dimension of $N(A)$ will be $n - r$. Similarly, the dimension of $N(A^T)$ will be $m - r$.

Let us illustrate this on the sample matrix in (2.47).

$$\begin{bmatrix} 1 & 2 & 3 & 1 \\ 2 & 3 & 5 & 2 \\ 3 & 4 & 7 & 3 \end{bmatrix} \overset{E_{2,1},E_{3,1}}{\Longrightarrow} \begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & -2 & -2 & 0 \end{bmatrix} \overset{E_{3,2}}{\Longrightarrow} (R =) \begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.49}$$

The reduced matrix R has the same row space as A, by virtue of the nature of row reduction. In fact, the rows of A can be retrieved from the rows of R by reversing the linear operations involved in row elimination. The first two rows give a basis for the row space of A. The dimension of $C(A^T)$ is 2, which is also the rank of A. To find the left null space of A, we look at the system $\mathbf{y}^T A = 0$. Recall the Gauss-Jordan elimination method from Section 2.4.2 that augments A with an $m \times m$ identity matrix, and performs row elimination on the augmented matrix.

$$[A\ I_{m\times m}] \overset{rref}{\Longrightarrow} [R\ E_{m\times m}]$$

The rref will consist of the reduced matrix augmented with the elimination matrix reproduced on its right. For the example case in 2.49, we apply the same elimination steps to obtain the matrix E below:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \overset{E_{2,1},E_{3,1}}{\Longrightarrow} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \overset{E_{3,2}}{\Longrightarrow} (E =) \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{2.50}$$

Writing down $EA = R$,

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 1 \\ 2 & 3 & 5 & 2 \\ 3 & 4 & 7 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.51}$$

We observe that the last row of E specifies a linear combination of the rows of A that yields a zero vector (corresponding to the last row of R). This is the only vector that yields a zero row in R and is therefore the only element in the basis of the left null space of A, that is, $N(A^T)$. The dimension of $N(A^T)$ is 1.

As another example, consider the space $\mathcal{S}$ of vectors $\mathbf{v} \in \Re^3$ where $\mathbf{v} = [v_1\ v_2\ v_3]^T$ such that $\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3 = \mathbf{0}$. What is the dimension of this subspace? Note that this subspace is the right null space $N(A)$ of a 1×3 matrix $A = [1\ 1\ 1]$, since $A\mathbf{v} = 0$. The rank, $r = rank(A)$ is 1, implying that the dimension of the right null space is $n - r = 3 - 1 = 2$. One set of basis vectors for $\mathcal{S}$ is $[-1\ 1\ 0]$, $[-1\ 0\ 1]$. The column space $C(A)$ is $\Re^1$ with dimension 1. The left null space $N(A^T)$ is the singleton set $\{0\}$ and as expected, has a dimension of $m - r = 1 - 1 = 0$.

2.8 Matrix Spaces

We will extend the set of examples of vector spaces discussed in Section 2.5 with a new vector space, that of all $m\times n$ matrices with real entries, denoted by $\Re^{m\times n}$.

It is easy to verify that the space of all matrices is closed under operations of addition and scalar multiplication. Additionally, there are interesting subspaces in the entire matrix space $\Re^{m\times n}$, *viz.*,

- set $\mathcal{S}$ of all $n \times n$ symmetric matrices
- set $\mathcal{U}$ of all $n \times n$ upper triangular matrices
- set $\mathcal{L}$ of all $n \times n$ lower triangular matrices
- set $\mathcal{D}$ of all $n \times n$ diagonal matrices

Let $\mathcal{M} = \Re^{3\times 3}$ be the space of all 3×3 matrices. The dimension of $\mathcal{M}$ is 9. Each element of this basis has a 1 in one of the 9 positions and the remaining entries as zeroes. Of these basis elements, three are symmetric (those having a 1 in any of the diagonal positions). These three matrices form the basis for the subspace of diagonal matrices. Six of the nine basis elements of M form the basis of $\mathcal{U}$ while six of them form the basis of $\mathcal{L}$.

The intersection of any two matrix spaces is also a matrix space. For example, $\mathcal{S} \cap \mathcal{U}$ is $\mathcal{D}$, the set of diagonal matrices. However the union of any two matrix spaces need not be a matrix space. For example, $\mathcal{S} \cup \mathcal{U}$ is not a matrix space; the sum $S + U,\ S \in \mathcal{S},\ U \in \mathcal{U}$ need not belong to $\mathcal{S} \cup \mathcal{U}$. We will discuss a special set comprising all linear combinations of the elements of union of two vector spaces $\mathcal{V}_1$ and $\mathcal{V}_2$ (*i.e.*, $\mathcal{V}_1 \cup \mathcal{V}_2$), and denote this set by $\mathcal{V}_1 \oplus \mathcal{V}_2$. By definition, this set is a vector space. For example, $\mathcal{S} + \mathcal{U} = \mathcal{M}$, which is a vector space.

A property fundamental to many properties of matrices is the expression for a rank 1 matrix. A rank 1 matrix can be expressed as the product of a column vector with a row vector (the row vector forming a basis for the matrix). Thus, any rank 1 matrix X can be expressed as

$$X_{m\times n} = u^T v = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ . \\ . \\ u_m \end{bmatrix} \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix} \tag{2.52}$$

Let $\mathcal{M}_{m\times n}$ be the set of all $m \times n$ matrices. Is the subset of $\mathcal{M}_{m\times n}$ matrices with rank k, a subspace? For $k = 1$, this space is obviously not a vector space as is evident from the sum of rank 1 matrices, A^1 and B^1, which is not a rank 1 matrix. In fact, the subset of $\mathcal{M}_{m\times n}$ matrices with rank k is not a subspace.

$$A^1 + B^1 = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 1 \\ 1 & 2 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 4 & 2 \\ 2 & 2 & 1 \\ 4 & 4 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 6 & 3 \\ 4 & 6 & 2 \\ 5 & 6 & 3 \end{bmatrix} \tag{2.53}$$

2.9 Orthogonality and Projection

In this section we will discuss the orthogonality of subspaces. Two vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be orthogonal *iff*, their dot product is 0. In the eucledian space, the dot product of the two vectors is x^Ty. The condition $x^Ty = 0$ is equivalent to the pythagorous condition between the vectors $\mathbf{x}$ and $\mathbf{y}$ that form the perpendicular sides of a right triangle with the hypotenuse given by $\mathbf{x}+\mathbf{y}$. The *pythagorous condition* is $||\mathbf{x}||^2 + ||\mathbf{y}||^2 = ||\mathbf{x}+\mathbf{y}||^2$, where the norm is the eucledian norm, given by $||\mathbf{x}||^2 = \mathbf{x}^T\mathbf{x}$. This equivalence can be easily proved and is left to the reader as an exercise. By definition, the vector $\mathbf{0}$ is orthogonal to every other vector.

We will extend the definition of orthogonality to subspaces; a subspace $\mathcal{U}$ is orthogonal to subspace $\mathcal{V}$ *iff*, every vector in $\mathcal{U}$ is orthogonal to every vector in $\mathcal{V}$. As an example:

Theorem 18 *The row space $C(A^T)$ of an $m \times n$ matrix A is orthogonal to its right null space $N(A)$.*

Proof: $A\mathbf{x} = \mathbf{0}$, $\forall \mathbf{x} \in N(A)$. On the other hand, $\forall\ \mathbf{y} \in C(A^T)$, $\exists\ z \in \Re^m$, $\mathbf{s}.t.$, $\mathbf{y} = A^T\mathbf{z}$. Therefore, $\forall\ \mathbf{y} \in C(A^T)$, $\mathbf{x} \in N(A), \mathbf{y}^T\mathbf{x} = \mathbf{z}^TA\mathbf{x} = \mathbf{z}.\mathbf{0} = 0$. $\Box$

Not only are $C(A^T)$ and the right null space $N(A)$ orthogonal to each other, but they are also *orthogonal complements* in $\Re^n$, that is, $N(A)$ contains all vectors that are orthogonal to some vector in $C(A^T)$.

Theorem 19 *The null space of A and its row space are orthogonal complements.*

Proof: We note, based on our discussion in Section 2.7.2 that the dimensions of the row space and the (right) null space add up to n, which is the number of columns of A. For any vector $\mathbf{y} \in C(A^T)$, we have $\exists\ \mathbf{z} \in \Re^m$, $\mathbf{s}.t.$, $\mathbf{y} = A^T\mathbf{z}$. Suppose $\forall\ \mathbf{y} \in C(A^T)$, $\mathbf{y}^T\mathbf{x} = 0$. That is, $\forall\ \mathbf{z} \in \Re^m$, $\mathbf{z}^TA\mathbf{x} = 0$. This is possible only if $A\mathbf{x} = \mathbf{0}$. Thus, necessarily, $\mathbf{x} \in N(A)$. $\Box$

Along similar lines, we could prove that the column space $C(A)$ and the left null space $N(A^T)$ are orthogonal complements in $\Re^m$. Based on theorem 19, we prove that there is a one-to-one mapping between the elements of row space and column space.

Theorem 20 *If $\mathbf{x} \in C(A^T)$, $\mathbf{y} \in C(A^T)$ and $\mathbf{x} \neq \mathbf{y}$, then, $A\mathbf{x} \neq A\mathbf{y}$.*

Proof: Note that $A\mathbf{x}$ and $A\mathbf{y}$ are both elements of $C(A)$. Next, observe that $\mathbf{x} - \mathbf{y} \in C(A^T)$, which by theorem 19, implies that $\mathbf{x} - \mathbf{y} \notin N(A)$. Therefore, $A\mathbf{x} - A\mathbf{y} \neq \mathbf{0}$ or in other words, $A\mathbf{x} \neq A\mathbf{y}$. $\Box$

Similarly, it can be proved that if $\mathbf{x} \in C(A)$, $\mathbf{y} \in C(A)$ and $\mathbf{x} \neq \mathbf{y}$, then, $A^T\mathbf{x} \neq A^T\mathbf{y}$. The two properties together imply a one-to-one mapping between the row and column spaces.

2.9.1 Projection Matrices

The projection of a vector $\mathbf{t}$ on a vector $\mathbf{s}$ is a vector $\mathbf{p} = c\mathbf{s},\ c \in \Re$ (in the same direction as $\mathbf{s}$), such that $\mathbf{t} - c\mathbf{s}$ is orthogonal to $\mathbf{s}$. That is, $\mathbf{s}^T(\mathbf{t} - c\mathbf{s}) = 0$ or $\mathbf{s}^T\mathbf{t} = c\mathbf{s}^T\mathbf{s}$). Thus, the scaling factor c is given by $c = \frac{\mathbf{s}^T\mathbf{t}}{\mathbf{s}^T\mathbf{s}}$. The projection of the vector $\mathbf{t}$ on a vector $\mathbf{s}$ is then

$$\mathbf{p} = \mathbf{s}\frac{\mathbf{t}^T\mathbf{s}}{\mathbf{s}^T\mathbf{s}} \tag{2.54}$$

Using the associative property of matrix multiplication, the expression for $\mathbf{p}$ can be re-written as

$$\mathbf{p} = P\mathbf{t} \tag{2.55}$$

where, $P = \mathbf{s}\mathbf{s}^T\frac{1}{\mathbf{s}^T\mathbf{s}}$ is called the *projection matrix.*

The rank of the projection matrix is 1 (since it is a column mutiplied by a row). The projection matrix is symmetric and its column space is a line through s. For any $d \in \Re$, $P(d\mathbf{s}) = d\mathbf{s}$, that is, the projection of any vector in the direction of $\mathbf{s}$ is the same vector. Thus, $P^2 = P$.

2.9.2 Least Squares

In Section 2.6.3, we saw a method for solving the system $A\mathbf{x} = \mathbf{b}$ (A being an $m \times n$ matrix), when a solution exists. Howevever, a solution may not exist, especially when $m > n$, that is when the number of equations is greater than the number of variables. In Section 2.6.3, we saw that the *rref* looks like $[I\ \mathbf{0}]^T$, where I is an $n \times n$ identity matrix. It could happen that the row reduction yields a zero submatrix in the lower part of A, but the corresponding elements in $\mathbf{b}$ are not zeroes. In other words, $\mathbf{b}$ may not be in the column space of A. In such cases, we are often interested in finding a 'best fit' for the system; a solution $\hat{x}$ that satisfies $A\mathbf{x} = \mathbf{b}$ as well as possible.

We define the best fit in terms of a vector $\mathbf{p}$ which is the projection of $\mathbf{b}$ onto $C(A)$ and solve $A\hat{\mathbf{x}} = \mathbf{p}$. We require that $\mathbf{b} - \mathbf{p}$ is orthogonal to $C(A)$, which means

$$A^T(\mathbf{b} - A\hat{\mathbf{x}}) = \mathbf{0} \tag{2.56}$$

The vector $\mathbf{e} = \mathbf{b} - A\hat{\mathbf{x}}$ is the error vector and is in $N(A^T)$. The equation (2.9.2) can be rewritten as

$$(A^T A)\hat{\mathbf{x}} = A^T\mathbf{b} \tag{2.57}$$

A matrix that plays a key role in this problem is A^TA. It is an $n \times n$ symmetric matrix (since $(A^TA)^T = A^TA$). The right null space $N(A^TA)$ is the same as $N(A)$[5]. It naturally follows that the ranks of A^TA and A are the same (since, the sum of the rank and dimension of null space equal n in either case). Thus, A^TA is invertible exactly if $N(A)$ has dimension 0, or equivalently, A is a full column rank.

Theorem 21 *If A is a full column rank matrix (that is, its columns are independent), A^TA is invertible.*

Proof: We will show that the null space of A^TA is $\{0\}$, which implies that the square matrix A^TA is full column (as well as row) rank is invertible. That is, if $A^TA\mathbf{x} = \mathbf{0}$, then $\mathbf{x} = \mathbf{0}$. Note that if $A^TA\mathbf{x} = \mathbf{0}$, then $\mathbf{x}^TA^TA\mathbf{x} = ||A\mathbf{x}|| = 0$ which implies that $A\mathbf{x} = \mathbf{0}$. Since the columns of A are linearly independent, its null space is $\mathbf{0}$ and therefore, $\mathbf{x} = \mathbf{0}$. $\square$

Assuming that A is full column rank, the equation (2.9.2) can be rewritten as

$$\hat{\mathbf{x}} = (A^TA)^{-1}A^T\mathbf{b}. \tag{2.58}$$

Therefore the expression for the projection $\mathbf{p}$ will be

$$\mathbf{p} = A(A^TA)^{-1}A^T\mathbf{b} \tag{2.59}$$

This expression is the n-dimensional equivalent of the one dimensional expression for projection in (2.9.1). The projection matrix in (2.59) is given by $P = A(A^TA)^{-1}A^T$. We will list the solution for some special cases:

- If A is an $n \times n$ square invertible matrix, its column space is the entire $\Re^n$ and the projection matrix will turn out to be the identity matrix.
- Also, if b is in the column space $C(A)$, then $\mathbf{b} = A\mathbf{t}$ for some $t\ in \Re^n$ and consequently, $P\mathbf{b} = A(A^TA)^{-1}(A^TA)\mathbf{t} = A\mathbf{t} = \mathbf{b}$.

[5]The proof is left as an exercise.

- On the other hand, if b is orthogonal to $C(A)$, it will lie in $N(A^T)$, and therefore, $A^T\mathbf{b} = 0$, implying that $\mathbf{p} = 0$.

Another equivalent way of looking at the best fit solution $\hat{\mathbf{x}}$ is a solution that minimizes the square of the norm of the error vector

$$e(\hat{\mathbf{x}}) = ||A\mathbf{x} - \mathbf{b}||^2 \tag{2.60}$$

Setting $\frac{de(\hat{\mathbf{x}})}{d\mathbf{x}} = 0$, we get the same expression for $\hat{\mathbf{x}}$ as in (2.9.2). The solution in 2.9.2 is therefore often called the *least squares solution*. Thus, we saw two views of finding a best fit; first was the view of projecting into the column space while the second concerned itself with minimizing the norm squared of the error vector.

We will take an example. Consider the data matrix A and the coefficient matrix $\mathbf{b}$ as in (2.61).

$$Ax = \begin{bmatrix} 2 & -1 \\ -1 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 3 \end{bmatrix} \tag{2.61}$$

The matrix A is full column rank and therefore A^TA will be invertible. The matrix A^TA is given as

$$A^TA = \begin{bmatrix} 6 & -3 \\ -3 & 6 \end{bmatrix}$$

Substituting the value of A^TA in the system of equations (2.9.2), we get,

$$6\hat{x}_1 - 3\hat{x}_2 = 2 - 3\hat{x}_1 + 6\hat{x}_2 = 8$$

The solution of which is, $x_1 = \frac{4}{5}$, $x_2 = \frac{26}{15}$.

2.9.3 Orthonormal Vectors

A collection of vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ is said to be orthonormal *iff* the following condition holds $\forall\ i, j$:

$$\mathbf{q}_i^T\mathbf{q}_j \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{2.62}$$

A large part of numerical linear algebra is built around working with orthonormal matrices, since they do not overflow or underflow. Let Q be a matrix comprising the columns $\mathbf{q}_1$ through $\mathbf{q}_n$. It can be easily shown that

$$Q^TQ = I_{n\times n}$$

When Q is square, $Q^{-1} = Q^T$. Some examples of matrices with orthonormal columns are:

$$Q_{rotation} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix},\ Q_{reflection} = \begin{bmatrix} cos(\theta) & sin(\theta) \\ sin(\theta) & -cos(\theta) \end{bmatrix},$$
$$Q_{Hadamard} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix},\ Q_{rect} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{2.63}$$

The matrix $Q_{rotation}$ when multiplied to a vector, rotates it by an angle θ, whereas $Q_{reflection}$ reflects the vector at an angle of $\theta/2$. These matrices present standard varieties of linear transformation, but in general, premultiplication by an $m \times n$ matrix transforms from an input space in $\Re^m$ to an input space in $\Re^n$. The matrix $Q_{Hadamard}$ is an orthonormal matrix consisting of only 1's and -1's. Matrices of this form exist only for specific dimensions such as 2, 4, 8, 16, *etc.*, and are called *Hadamard matrices*[6]. The matrix Q_{rect} is an example rectangular matrix whose columns are orthonormal.

Suppose a matrix Q has orthonormal columns. What happens when we project any vector onto the column space of Q? Substituting $A = Q$ in (2.59), we get[7]:

$$\mathbf{p} = Q(Q^TQ)^{-1}Q^T\mathbf{b} = QQ^T\mathbf{b} \tag{2.64}$$

Making the same substitution in (2.9.2),

$$\hat{\mathbf{x}} = (A^TQ)^{-1}Q^T\mathbf{b} = Q^T\mathbf{b} \tag{2.65}$$

The i^{th} component of $\mathbf{x}$, is given by $x_i = q_i^Tb$.

Let Q_1 be one orthonormal basis and Q_2 be another orthonormal basis for the same space. Let A be the coefficient matrix for a set of points represented using Q_1 and B be the coefficient matrix for the same set of points represented using Q_2. Then $Q_1A = Q_2B$, which implies that B can be computed as $B = Q_2^TQ_1A$. This gives us the formula for changing basis.

[6]An exhaustive listing of different types of matrices can be found at `http://en.wikipedia.org/wiki/List_of_matrices`.

[7]Note that $Q^TQ = I$. However, $QQ^T = I$ only if Q is a square matrix.

2.9.4 Gram-Schmidt Orthonormalization

The goal of the Gram-Schmidt orthonormalization process is to generate a set of orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$, given a set of independent vectors $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$. The first step in this process is to generate a set of orthogonal vectors $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$ from $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$. To start with, $\mathbf{t}_1$ is chosen to be $\mathbf{a}_1$. Next, the vector $\mathbf{t}_2$ is obtained by removing the projection of $\mathbf{a}_2$ on $\mathbf{t}_1$, from $\mathbf{a}_2$, based on (2.9.1). That is,

$$\mathbf{t}_2 = \mathbf{a}_2 - \frac{1}{\mathbf{a}_1^T\mathbf{a}_1}\mathbf{a}_1\mathbf{a}_1^T\mathbf{a}_2 \tag{2.66}$$

This is carried out iteratively for $i = 1, 2, \ldots, n$, using the expression below:

$$\mathbf{t}_i = \mathbf{a}_i - \frac{1}{\mathbf{t}_1^T\mathbf{t}_1}\mathbf{t}_1\mathbf{t}_1^T\mathbf{a}_i - \frac{1}{\mathbf{t}_2^T\mathbf{t}_2}\mathbf{t}_2\mathbf{t}_2^T\mathbf{a}_i - \ldots - \frac{1}{\mathbf{t}_{i-1}^T\mathbf{t}_{i-1}}\mathbf{t}_{i-1}\mathbf{t}_{i-1}^T\mathbf{a}_i \tag{2.67}$$

This gives us the orthogonal vectors $\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_n$. Finally, the orthonormal vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ are obtained by the simple expression

$$\mathbf{q}_i = \frac{1}{||\mathbf{t}_i||}\mathbf{t}_i \tag{2.68}$$

Let A be the matrix with columns $\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$ and Q, the matrix with columns $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$. It can be proved that $C(V) = C(Q)$, that is, the matrices V and Q have the same column space. The vector a_i can be expressed as

$$\mathbf{a}_i = \sum_{k=1}^{n}(\mathbf{a}_i^T\mathbf{q}_k)\mathbf{q}_k \tag{2.69}$$

The i^{th} column of A is a linear combination of the columns of Q, with the scalar coefficient $\mathbf{a}_i^T\mathbf{q}_k$ for the k^{th} column of Q. By the very construction procedure of the Gram-Schmidt orthonormalization process, $\mathbf{a}_i$ is orthogonal to $\mathbf{q}_k$ for all $k > i$. Therefore, (2.69) can be expressed more precisely as

$$\mathbf{a}_i = \sum_{k=1}^{i}(\mathbf{a}_i^T\mathbf{q}_k)\mathbf{q}_k \tag{2.70}$$

Therefore, matrix A can be decomposed into the product of Q with a upper triangular matrix R; $A = QR$, with $R_{k,i} = \mathbf{a}_i^T\mathbf{q}_k$. Since $\mathbf{a}_i^T\mathbf{q}_k = 0$, $\forall\ k > i$, we can easily see that R is upper traingular.

2.9.5 Fourier and Wavelet Basis

The hermetian inner product of a complex vector $\mathbf{x}$ with another complex vector $\mathbf{y}$ is $\overline{\mathbf{x}}^T\mathbf{y}$, and is also denoted by $\mathbf{x}^H\mathbf{y}$. A complex matrix Q is called orthonormal if $\overline{Q}^T Q = I$. Consider the complex number $c = cos(\frac{2\pi}{n}) + icos(\frac{2\pi}{n})$. Then, $w^n = 1$. The *fourier matrix* F_n is defined as

$$F_n = \frac{1}{n}\begin{bmatrix} 1 & 1 & \ldots & 1 \\ 1 & c & \ldots & c^{n-1} \\ . & . & \ldots & . \\ 1 & c^{k-1} & \ldots & c^{(k-1)(n-1)} \\ . & . & \ldots & , \\ 1 & c^{n-1} & \ldots & c^{(n-1)(n-1)} \end{bmatrix} \tag{2.71}$$

The (hermetian) inner products of distinct columns F_n are 0, while the inner product of a column with itself is 1. Therefore, the columns of F_n are orthonormal and form a basis for $\Re^n$. Consequently, the inverse of F_n is its conjugate transpose $\overline{F}_n^T$.

Further, $F_{2^k}, k \geq 1$ can expressed as

$$F_{2^k} == \begin{bmatrix} I & D \\ I & -D \end{bmatrix}\begin{bmatrix} F_{2^{k-1}} & 0_{2^{k-1}} \\ 0_{2^{k-1}} & F_{2^{k-1}} \end{bmatrix}\underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & 1 & \ldots & 0 & 0 \\ . & . & . & . & \ldots & . & . \\ 0 & 0 & 0 & 0 & \ldots & 0 & 1 \\ 1 & 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 1 & 0 & \ldots & 0 & 0 \\ . & . & . & . & \ldots & . & . \\ 0 & 0 & 0 & 0 & \ldots & 1 & 0 \end{bmatrix}}_{P} \tag{2.72}$$

where, $D = diag(1, c, c^2, \ldots, c^{2^k})$ and 0_{2^k} is a $2^k \times 2^k$ matrix of 0's. This factorization, applied recursively, can reduce the time for computing F_{2^k} from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$. This is the idea behind the *fast fourier transform algorithm.* Though the factorization discussed here, applies to only to powers of 2, there exist FFT algorithms [**?**] for any number (including primes).

An advantage of representing a vector in $\Re^n$ (for example, a $\sqrt{n} \times \sqrt{n}$ sub-block of an image matrix) using the fourier basis is that certain basis components of the representation could be ignored to achieve minimally lossy compression of matrices such as image matrices. Another orthogonal basis that is used for minimally lossy matrix compression is the wavelet basis. A sample wavelet basis matrix W for $\Re^8$ is

$$W = \begin{bmatrix} 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & -1 & 0 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \\ 1 & -1 & 0 & -1 & 0 & 0 & 0 & -1 \end{bmatrix} \tag{2.73}$$

The discrete wavelet transform can be computed efficiently using a *fast wavelet transform algorithm* which is less computationally complex, taking $\mathcal{O}(n)$ time as compared to $\mathcal{O}(nlogn)$ for the fast fourier transform.

2.10 Determinants

Every square matrix A has a real number associated with it, called its determinant and it is denoted by $det(A)$. In this sequel, we will often refer to the following $n \times n$ matrix A:

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{k1} & a_{k2} & \ldots & a_{kn} \\ . & . & \ldots & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \tag{2.74}$$

We will describe four fundamental properties of the determinant, which essentially define the determinant.

1. The determinant of the identity matrix I is 1. That is, $det(I) = 1$.
2. When two rows of A are permuted (*c.f.* Section 2.3.1), the sign of the determinant changes. That is $det\left(Perm(A, j, k)\right) = -det(A)$, where $Perm(A, j, k)$ returns a matrix formed by exchanging the j^{th} and k^{th} rows of A for any $1 \leq j, k \leq n$.
3. If any row of A is scaled by $t \in \Re$, the determinant also gets scaled by t. Thus, if

$$S(A,k,t) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ ta_{k1} & ta_{k2} & \dots & ta_{kn} \\ . & . & \dots & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{2.75}$$

then

$$det\,(S(A,k,t)) = t \times det(A) \tag{2.76}$$

The function $S(A,k,t)$ returns a matrix eaxactly with all the entries of A, except for the k^{th} row, which is scaled by $t \in \Re$.

4. The sum of the determinants of two $n \times n$ matrices, A and B, with all $(n-1)$ rows the same, except for the k^{th} row, $1 \leq k \leq n$, equals the determinant of an $n \times n$ matrix C that has the same $n-1$ rows from A/B, but with the k^{th} row being the sum of the k^{th} rows of A and B. Thus, if

$$B = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ b_{k1} & b_{k2} & \dots & b_{kn} \\ . & . & \dots & . \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{2.77}$$

and,

$$C = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ . & . & \dots & . \\ a_{k1}+b_{k1} & a_{k2}+b_{k2} & \dots & a_{kn}+b_{kn} \\ . & . & \dots & , \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{2.78}$$

then

$$det(C) = det(A) + det(B)$$

Using these basic properties of determinants, we infer some **derived properties**:

1. If a matrix A has two equal rows, its determinant must be 0.

   Proof: Let B be the matrix obtained by permuting the two equal rows of A. By the second property of determinants, $det(B) = -det(A)$. Since the permuted rows are the same, $B = A$, which implies that $det(B) = det(A)$. The two equalities on determinants of A and B imply that $det(A) = 0$. □

2. The determinant of A obtained by subtracting $\rho \in \Re$ times the j^{th} row from the k^{th} row leaves the determinant unaltered. Therefore, if

$$E = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{k1} - \rho a_{j1} & a_{k2} - \rho a_{j2} & \ldots & a_{kn} - \rho a_{jn} \\ . & . & \ldots & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \tag{2.79}$$

we will have

$$det(E) = det(A)$$

Proof:

$$det(E) = det(A) + det\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ -\rho a_{j1} & -\rho a_{j2} & \ldots & -\rho a_{jn} \\ . & . & \ldots & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}\right)$$

$$= det(A) - \rho \times det\left(\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ . & . & \ldots & . \\ a_{j1} & a_{j2} & \ldots & a_{jn} \\ . & . & \ldots & . \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}\right) = det(A) \tag{2.80}$$

The first step follows from the fourth fundamental property of determinants, the second follows from the third fundamental property, while the third step is a consequence of the first derived property of determinants. □

This property shows that the row elimination steps discussed in Section 2.3.1, leave the determinant unchanged.

3. If any k^{th} row of A is 0, then $det(A) = 0$ *Proof:* Consider a matrix A' that has the same rows as A for all $1 \leq i \leq n$, except for the $i = k$. Let the k^{th} row of A' be the same as its j^{th} row, for some $1 \leq j \leq n$, such that $j \neq k$. Note that by the first derived property, $det(A') = 0$. The matrix A can be obtained from A' by subtracting the j^{th} row of A' from its k^{th} row. By the second derived property, $det(A) = det(A')$. Thus, $det(A) = 0$. Another simpler proof is that $S(A, k, 0) = A$ which implies that $det(A) = det\left(S(A, k, 0)\right) = 0 \times det(A) = 0$ (by third fundamental property of determinants). □

4. The determinant of an upper triangular matrix U is the product of its diagonal entries.

   Proof: Row elimination operations can be performed on an upper traingular matrix U to yield a diagonal matrix D (*c.f.* Section 2.4.2 on Gauss-Jordan elimination), while neither performing any row exchanges nor altering the diagonal entries of U. By the second derived property of determinants, $det(D) = det(U)$. Using the first fundamental property of determinants, $det(D)$ can be proved to be the product of its diagonal entries, which is also the product of the diagonal entries of U. □

   In fact, most mathematical softwares compute the determinant of a matrix A by first reducing it to an upper triangular matrix U by row elimination on A (which preserves the determinant, by virtue of the second derived property) and then compute the product of the diagonal entries (which also happen to be the pivots) of U. If some α row exchanges are performed during the reduction of A to U, the product of the diagonal entries, multiplied by $(-1)^{\alpha}$ yields the determinant of A. As an example, consider the 2×2 matrix A_2:

$$A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{2.81}$$

   Using the derived property (2), the matrix $A_2^{'}$ can proved to have same determinant as A.

$$A_2^{'} = \begin{bmatrix} a_{11} & a_{12} \\ 0 & a_{22} - \frac{a_{21} * a_{12}}{a_11} \end{bmatrix} \tag{2.82}$$

$A_2^{'}$ is an upper triangular matrix with $det(A_2^{'})$ given by

$$det(A_2^{'}) = a_{11}a_{22} - a_{21} * a_{12} \tag{2.83}$$

Therefore,

$$det(A) = a_{11}a_{22} - a_{21} * a_{12} \tag{2.84}$$

5. The determinant of a matrix A is 0 *iff* A is singular (or non-invertible).

   Proof Sketch: We will consider the proof in two parts. When A is singular, elimination, with some possible permutations, yields (as discussed in Section 2.4.1) an upper traingular matrix with some diagonal (pivot) entries that are 0s. Therefore, by the derived property (4), $det(A) = 0$. When A is non-singular, elimination yields an upper triangular matrxi with no zero entries. Consequently, we will have $det(A) \neq 0$. $\square$

6. The determinant of the product of two matrices A and B is the product of their determinants[8], *i.e.*, $det(AB) = det(A) \times det(B)$.

   A corollary of this property is that $det(A^{-1}) = \frac{1}{det(A)}$ because $det(A^{-1})det(A) = det(I) = 1$. Similarly, $det(A^n) = (det(A))^n$ and $det(A_1A_2\ldots A_n) = det(A_1)det(A_2)\ldots det(A_n)$.

   In this context, it will be appropriate to point out that determinants also have relationship with volumes of solids. The determinant of matrix A in (2.74), is the volume of an $n-$dimensional parallelotope, with corners at $(0, 0, \ldots, 0)$, $(a_{11}, a_{12}, \ldots, a_{1n})$, …, $(a_{n1}, a_{n2}, \ldots, a_{nn})$. The parallelotope corresponding to $I_{n\times n}$ is an $n-$dimensional unit hypercube in n dimensions and has a volume of 1. An orthonormal matrix Q represents a hypercube in n dimensions and has volume given by $det(Q) = \sqrt{det(I)} = 1$. If Q is orthogonal (and not necessarily orthonormal), its volume is $\prod_{i=1}^{n} s_i$, where s_i is the factor by which the i^{th} row of Q should be scaled, so that the row has unit norm. Determinants make easy the task of computing areas of parallelotopes. If the parallelotope does not have any corner at the origins, the coordinates of the corners can be computed relative any one of the corners and the area can be computed using determinants.

7. The determinant of a square matrix A equals the determinant of its transpose, *i.e.*, $det(A) = det(A^T)$.

[8]Recall that determinant does not have the linear additive property.

Proof Sketch: We can decompose A as a LU, where L is a lower traingular matrix and U is an upper traingular matrix. That is $A = LU$. Consequently, $A^T = U^T L^T$. By derived property (6), $det(A) = det(L)det(U)$ and $det(A^T) = det(U^T)det(L^T)$. Since the diagonal entries of L^T and U^T are the same as the diagonal entries of L and U respectively, and since derived property (4) states that the determinants of L, L^T, U and U^T are just products of their respective diagonal entries, we have $det(A) = det(A^T)$. $\Box$

By virtue of this property, all the properties of determinants discussed so far with respect to scaling or exchanging rows hold for similar manipulations on the columns, since column operations on A are row operations on A^T.

2.10.1 Formula for determinant

In (2.84), we showed the formula for the determinant of a 2×2 matrix A_2. The formula can also be obtained by using the basic property (4), decomposing $det(A_2)$ into the sum of determinants of 4 matrices, with one surviving element per row. We will use the notation $|.|$ instead of det ([.]) to denote the determinant of a matrix.

$$det(A_2) = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = \begin{vmatrix} a_{11} & 0 \\ a_{21} & 0 \end{vmatrix} + \begin{vmatrix} a_{11} & 0 \\ 0 & a_{22} \end{vmatrix} + \begin{vmatrix} 0 & a_{12} \\ a_{21} & 0 \end{vmatrix} + \begin{vmatrix} 0 & a_{12} \\ 0 & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21} \tag{2.85}$$

Of these, there are only two nonzero terms; the terms with zero columns or zero rows have 0 determinant. The determinant of a 3×3 matrix can be similarly computed, by decomposing the determinant as the sum of $3 \times 3 \times 3 = 27$ determinants. However, many of the determinants in the sum turn out to be 0, either because of zero rows or zero columns. Each of the non-zero terms have exactly one entry for each row and each column. Thus, the determinant of a 3×3 matrix can be expressed as

$$\begin{aligned}
det(A_3) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} &= \begin{vmatrix} a_{11} & 0 & 0 \\ 0 & a_{22} & 0 \\ 0 & 0 & a_{33} \end{vmatrix} + \begin{vmatrix} a_{11} & 0 & 0 \\ 0 & 0 & a_{23} \\ 0 & a_{32} & 0 \end{vmatrix} \\
&+ \begin{vmatrix} 0 & a_{12} & 0 \\ 0 & 0 & a_{23} \\ a_{31} & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & a_{12} & 0 \\ a_{21} & 0 & 0 \\ 0 & 0 & a_{33} \end{vmatrix} \\
&+ \begin{vmatrix} 0 & 0 & a_{13} \\ 0 & a_{22} & 0 \\ a_{31} & 0 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & a_{13} \\ a_{21} & 0 & 0 \\ 0 & a_{32} & 0 \end{vmatrix} \\
= a_{11}a_{22}a_{33} - a_{11}a_{23}a_{32} &+ a_{12}a_{23}a_{31} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31} + a_{13}a_{21}a_{32} \\
= a_{11}\left(a_{22}a_{33} - a_{23}a_{32}\right) &- a_{12}\left(a_{21}a_{33} - a_{23}a_{31}\right) + a_{13}\left(a_{21}a_{32} - a_{22}a_{31}\right) \\
= a_{11} \underbrace{\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}}_{cofactor\ of\ a_{11}} &+ \underbrace{a_{12}(-1) \underbrace{\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}}_{minor\ of\ a_{12}}}_{cofactor\ of\ a_{12}} + a_{13} \underbrace{\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}}_{cofactor\ of\ a_{13}}
\end{aligned} \tag{2.86}$$

In (2.86), the determinant of A_3 is decomposed into the sum of signed determinants of smaller 2×2 matrices called *co-factors*, each scaled by a corresponding *factor*. The sign of the co-factors depend on the number of row permutations required to get the matrix in a diagonal form; the sign is $(-1)^{num\ perms}$ which happens to be $(-1)^{i+j}$. In general, for an $n \times n$ matrix A, the *minor* of a term a_{ij} is the determinant of an $(n-1) \times (n-1)$ sub-matrix of A that has the row i and column j removed, while its *co-factor* is the minor multiplied by $(-1)^{i+j}$. The minor for a_{ij} is denoted by M_{ij} and its co-factor by C_{ij}. Minors of the form M_{ii} are called principal minors.

The general formula for the determinant of an $n \times n$ matrix contains $n!$ terms, corresponding to all permutations of the choice of the column index for the non-zero entries corresponding to each row index. That is,

$$det(A) = \sum_{(p_1,p_2,\ldots,p_n) \in Perm(1,2,\ldots,n)} a_{1p_1}a_{2p_2}\ldots a_{p_n} \tag{2.87}$$

In terms of co-factors, the formula for determinant is

$$det(A) = \sum_{k=1}^{n} a_{ik}C_{ik} \tag{2.88}$$

for any $1 \le i \le n$.

2.10.2 Formula for Inverse

Let A be an $n \times n$ invertible matrix. In Section 2.4.1, we saw an elegant algorithm for computing A^{-1}. In (2.89), we present a closed form expression for A^{-1} in terms of the co-factors of A, even though the expression is very expensive to compute.

$$A^{-1} = \frac{1}{det(A)} \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1n} \\ C_{21} & C_{22} & \dots & C_{2n} \\ . & . & \dots & . \\ C_{k1} & C_{k2} & \dots & C_{kn} \\ . & . & \dots & . \\ C_{n1} & C_{n2} & \dots & C_{nn} \end{bmatrix}^T = \frac{1}{det(A)} C^T \tag{2.89}$$

We denote the matrix in (2.89) consisting of the co-factors of A, by C. It can be easily verified that the expression in (2.89) is indeed A^{-1}.

$$AA^{-1} = \frac{1}{det(A)} \begin{bmatrix} \sum_{j=1}^{n} a_{1j}C_{1j} & \sum_{j=1}^{n} a_{1j}C_{2j} & \dots & \sum_{j=1}^{n} a_{1j}C_{nj} \\ \sum_{j=1}^{n} a_{2j}C_{1j} & \sum_{j=1}^{n} a_{2j}C_{2j} & \dots & \sum_{j=1}^{n} a_{2j}C_{nj} \\ . & . & \dots & . \\ . & . & \sum_{j=1}^{n} a_{ij}C_{ij} & . \\ . & . & \dots & . \\ \sum_{j=1}^{n} a_{nj}C_{1j} & \sum_{j=1}^{n} a_{nj}C_{2j} & \dots & \sum_{j=1}^{n} a_{nj}C_{nj} \end{bmatrix}$$

$$= \frac{1}{det(A)} \begin{bmatrix} det(A) & 0 & \dots & 0 \\ 0 & det(A) & \dots & 0 \\ . & . & \dots & . \\ . & . & det(A) & . \\ . & . & \dots & . \\ 0 & 0 & \dots & det(A) \end{bmatrix} = I_{n \times n} \tag{2.90}$$

Recall from (2.88) that $det(A) = \sum_{j=1}^{n} a_{ij}C_{ij}$ for any $1 \leq i \leq n$. However, $\sum_{j=1}^{n} a_{ij}C_{kj} = 0$, if $i \neq k$. This is because, for $i \neq k$, $\sum_{j=1}^{n} a_{ij}C_{kj}$ is the determinant of a matrix that has identical i^{th} and k^{th} rows and hence equals 0, by the derived property (1) for matrices.

The formula (2.89) for matrix inverse (if it exists) can be substituted in (2.4.1) to yield the *Cramer's rule* for solving the system $A\mathbf{x} = \mathbf{b}$. The Cramer's rule is:

$$
\begin{aligned}
x = A^{-1}\mathbf{b} &= \frac{1}{det(A)} \begin{bmatrix} \sum_{j=1}^{n} b_j C_{1j} \\ \sum_{j=1}^{n} b_j C_{2j} \\ . \\ . \\ \sum_{j=1}^{n} b_j C_{nj} \end{bmatrix} \\
&= \frac{1}{det(A)} \begin{bmatrix} det(B_1) \\ det(B_2) \\ . \\ . \\ det(B_n) \end{bmatrix} \qquad (2.91)
\end{aligned}
$$

B_i is a matrix obtained by replacing the i^{th} column of A with the vector $\mathbf{b}$ and keeping all other columns unaltered. This rule is never used in practical computations; the explicit formula only helps in analysis or derivation.

2.11 Eigenvalues and Eigenvectors

Let A be an $n \times n$ square matrix. Consider the function $f : \Re^n \rightarrow \Re^n$, defined as $f(\mathbf{x}) = A\mathbf{x}$. Suppose we are interested in vectors $\mathbf{x}$, for which, f returns a vector in the same direction as $\mathbf{x}$. Such vectors are called *eigenvectors* of A.

Eigenvector: *Vector* $\mathbf{x} \in \Re^n$ *is called an eigenvector of an* $n \times n$ *matrix* A*, iff*

$$A\mathbf{x} = \lambda\mathbf{x}, \ \exists \ \lambda \in \Re \qquad (2.92)$$

The scalar λ *is called an* **eigenvalue** *of* A*, corresponding to the eigenvector* $\mathbf{x}$.

We will consider some special examples of eigenvectors and eigenvalues.

- For the simple case of $\lambda = 0$, any $\mathbf{x} \in N(A)$ is an eigenvalue of A. Thus, if A is singular, so that $N(A) \neq \{\}$, $\lambda = 0$ and $\mathbf{x} \in N(A)$ are a valid eigenvalue-eigenvector pair.

- If A happens to be a projection matrix, (*c.f.*, Section 2.9.1), *i.e.*,

$$A = \mathbf{s}\mathbf{s}^T \frac{1}{\mathbf{s}^T\mathbf{s}}$$

  for some $\mathbf{s} \in \Re^n$. Recall that, $A\mathbf{x} = \mathbf{x}$ for any $\mathbf{x}$ in the column space of A. Therefore, $\mathbf{x} = \rho\mathbf{s}$ is an eigenvector of A for any $\rho \in \Re$, with 1 as the corresponding eigenvalue. As discussed above, any $\mathbf{x} \in N(A)$ is also an eigenvector of A with a corresponding eigenvalue of 0. However, for any other $\mathbf{x} \notin C(A)$, $A\mathbf{x} = c\mathbf{s}$ and therefore x is not an eigenvector.

Consider the permutation matrix P_{23} in (2.93):

$$P_{23} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.93}$$

By inspection, we find that P_{23} has atleast three eigenvectors, *viz.*, $x_1 = [1 \quad 0 \quad 0]$ with eigenvalue $\lambda_1 = 1$, $x_2 = [0 \quad 1 \quad 1]$ with eigenvalue $\lambda_2 = 1$, and $x_3 = [0 \quad -1 \quad 1]$ with eigenvalue $\lambda_3 = -1$. Does P_{23} have any more eigenvectors? The answer is no. It turns out that any $n \times n$ matrix has exactly n orthonormal eigenvectors. Moreover, the trace of a matrix (*i.e.*, the sum of its diagonal entries) always equals the sum of the eigenvalues corresponding to the orthonormal eigenvectors.

$$tr(A) = \sum_{i=1}^{n} \lambda_i$$

Thus, if we knew $n-1$ eigenvalues of a matrix, we could easily determine its n^{th} eigenvalue. We will defer this discussion to a later part of this chapter.

2.11.1 Solving for Eigenvalues

The equation (2.92) defining the criterion for an eigenvalue $\mathbf{x}$ can we re-written as in (2.94).

$$(A - \lambda I)\mathbf{x} = \mathbf{0} \tag{2.94}$$

For a solution $\mathbf{x}$ to exist, $A - \lambda I$ must be singular (*i.e.*, non-invertible) and $\mathbf{x}$ must lie in the null space $N(A - \lambda I)$. Therefore, $det(A - \lambda I) = 0$ is a necessary and sufficient condition for λ to be an eigenvalue. Once the eigenvalue λ is determined, the corresponding eigenvectors can be determined by computing $N(A - \lambda I)$, a procedure that has been already discussed in Section 2.5.2. We will therefore first discuss the procedure for computing the solution to

$$det(A - \lambda I) = 0 \tag{2.95}$$

As an example, when we apply the criterion in (2.95), to the matrix P_{23}, we get solutions as shown in (2.96):

$$\begin{aligned} det(P_{23} - \lambda I) &= (1 - \lambda)\lambda^2 = 0 \\ &\Rightarrow \lambda = 1 \textit{ or } \lambda = -1 \end{aligned} \tag{2.96}$$

Substituting these two values into the system $(A - \lambda I)\mathbf{x} = \mathbf{0}$, we get one matrix for each possible value of λ. It can be verified that the basis for the null space of $(A-\lambda I)$ obtained using the elimination process discussed in Section 2.6 (particularly, equation 2.27) is indeed $[1 \ \ 0 \ \ 0]^T$ and $[0 \ \ 1 \ \ 1]^T$ for eigenvalue $\lambda_1 = 1$, and $[0 \ \ -1 \ \ 1]$ for eigenvalue $\lambda_3 = -1$.

2.11.2 Some Properties of Eigenvalues and Eigenvectors

How are the eigenvectors and eigenvalues of a matrix affected when transformations are performed on the matrix? Below, we list some properties of eigenvalues with respect to matrix transformations.

1. If $A\mathbf{x} = \lambda\mathbf{x}$, then $(A + kI)\mathbf{x} = (\lambda + k)\mathbf{x}$. That is, the eigenvalues of $A + \lambda I$ are the eigenvalues of A, incremented by k, without any change in corresponding eigenvectors.

2. Consider the matrix R in (2.97):

$$R = \begin{bmatrix} 0 & 3 \\ -2 & 0 \end{bmatrix} \tag{2.97}$$

   The eigenvalues of R can be found as follows: $det(R - \lambda I) = \lambda^2 + 6 = 0 \Rightarrow \lambda = \pm\sqrt{6}i$. The eigenvalues of a matrix could be complex numbers as this example illustrates. In fact, eigenvalues always appear as complex conjugates, as in this example.

3. Let λ be an eigenvalue of A and $\mathbf{x}$ its corresponding eigenvector, *i.e.*, $A\mathbf{x} = \lambda\mathbf{x}$. It can be shown that the complex conjugates $\overline{\lambda}$ and $\overline{\mathbf{x}}$ also form an eigenvalue-eigenvector pair for $\overline{A}$. Thus, $\overline{A}\overline{\mathbf{x}} = \overline{\lambda}\overline{\mathbf{x}}$. If A happens to have only real entries, then, $A\overline{\mathbf{x}} = \overline{\lambda}\overline{\mathbf{x}}$.

4. The eigenvalues of upper and lower traingular matrices can be computed very easily. By derived property (4) of determinants, the determinant of an upper traingular matrix is the product of its diagonal entries. Let U be an $n \times n$ upper traingular matrix, with u_{ij} being the entry corresponding to the i^{th} row and j^{th} column. First we note that $U - \lambda I$ will also be upper traingular, since I is upper traingular and since the sum of upper traingular matrices is also upper traingular (the space of upper traingular matrices is a vector space, as shown in Section 2.8). Now, $det(U - \lambda I) = \prod_{i=1}^{n}(u_{ii} - \lambda) = 0$. The eigenvalues correspond to solutions of this equation; they are $\lambda_i = u_{ii}$, $1 \leq i \leq n$. The eigenvectors can be computed by solving the systems $(U - \lambda_i I)\mathbf{x}_i = \mathbf{0}$ by simple back-subtitutions, as illustrated in Section 2.3.1.

5. If $\mathbf{x}$ is an eigenvector of A with a corresponding eigenvalue λ, we have $A\mathbf{x} = \lambda\mathbf{x}$. Therefore, $A^2\mathbf{x} = A(A\mathbf{x}) = \lambda A\mathbf{x} = \lambda^2\mathbf{x}$. Thus, $\mathbf{x}$ is an eigenvector of A^2 as well, with a corresponding eigenvalue of λ^2. This statement can be generalized: *If $\mathbf{x}$ is an eigenvector of A with a corresponding eigenvalue λ, $\mathbf{x}$ is also an eigenvector of A^k, with corresponding eigenvector λ^k.*

6. The eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of a matrix A are linearly independent if all its eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ are different. This can be proved by contradiction[9] However, the eigenvectors, could be independent even if eigenvalues are repeated; but it is not always true. For instance, any traingular matrix having some identical diagonal elements (as in the case of the identity matrix) has linearly independent eigenvectors, even though some eigenvalues are identical.

7. In many engineering problems, we are faced with the system of equations

$$\mathbf{b}_{i+1} = A\mathbf{b}_i, \ \forall \ i \geq 0 \tag{2.98}$$

That is, $\mathbf{b}_i = A^i\mathbf{b}_0$. If A has n linearly independent eigenvectors (so that they span $\Re^n$), these systems can be solved efficiently, by expressing $\mathbf{b}_0$ as a linear combination of the eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ of A.

$$\mathbf{b}_0 = \sum_{k=1}^{n} c_k \mathbf{v}_k$$

[9]Exercise.

where, $c_k \in \Re$, $\forall\, 1 \leq k \leq n$. Consequently, any $\mathbf{b}_i,\ i \geq 0$ can be computed efficiently as

$$\mathbf{b}_i = \sum_{i=k}^{n} \lambda_k^i c_k \mathbf{v}_k \tag{2.99}$$

8. Consider the fibonacci sequence $f_{i+2} = f_i + f_{i+1} i \geq 0$, with $f_0 = 0$ and $f_1 = 1$. The recurrance relation can be written as a linear system (2.100).

$$\underbrace{\begin{bmatrix} f_{i+2} \\ f_{i+1} \end{bmatrix}}_{\mathbf{b}_{i+1}} = \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} f_{i+1} \\ f_i \end{bmatrix}}_{\mathbf{b}_i} \tag{2.100}$$

Note that $\mathbf{b}_0 = [0\ \ 1]^T$. The system of equations (2.100) is of the same form $\mathbf{b}_{i+1} = A\mathbf{b}_i$, $\forall\, 0 \leq i \leq n$ discussed above and therefore, the expression for $\mathbf{b}_i$ can be derived using (2.99), after computing values of λ_1, $\mathbf{v}_1$, c_1 and λ_2, $\mathbf{v}_2$ and c_2. The values of $\lambda_1 = \frac{1}{2}(1+\sqrt{5}) = 1.6180$ and $\lambda_2 = \frac{1}{2}(1-\sqrt{5}) = -0.6180$ can be computed by solving $det(A - \lambda I) = 0$. Substituting these values of λ, eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ can be obtained as in (2.101).

$$\mathbf{v}_1 = \begin{bmatrix} -0.8507 \\ -0.5257 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0.5257 \\ -0.8507 \end{bmatrix} \tag{2.101}$$

A closed form expression is $\mathbf{b}_i = c_1(1.6180)^i[-0.8507\ \ -0.525]^T - c_2(0.6180)^i[0.525\ -0.8507]^T$.

Another application of this general technique is in differential equations. Let us say we are given the differential equation $x'' + a_1 x' + a_2 x = 0$. This equation can be equivalently expressed as

$$\mathbf{y}' = \underbrace{\begin{bmatrix} -a_1 & -a_2 \\ 1 & 0 \end{bmatrix}}_{A} \mathbf{y} \tag{2.102}$$

where, $\mathbf{y} = [x'\ \ x]^T$. The n^{th} derivative of x can expressed in a closed form by determining the eigenvalues and eigenvectors of A.

9. If λ is an eigenvalue of a matrix A, then it is also an eigenvalue of A^T. This is because $det(A - \lambda I) = det\left((A - \lambda I)^T\right) = det(A^T - \lambda I)$. The eigenvectors for the same eigenvalues could however differ between A and A^T.

10. Another general property of any square matrix is that the sum of its eigenvalues equals its trace. Additionally, the product of its eigenvalues equals its determinant. Consequently, for any 2×2 matrix, if the trace is negative and the determinant positive, the real parts of both its eigenvalues must be negative.

11. A Markov matrix[10] M is an $n \times n$ matrix such that (1) all its entries are ≥ 0 and (2) the sum of all entries in each column is 1. An example Markov matrix M_3 is

$$M = \begin{bmatrix} 0.1 & 0.25 & 0.3 & 0.35 \\ 0.2 & 0.25 & 0.3 & 0.05 \\ 0.3 & 0.25 & 0.4 & 0.15 \\ 0.4 & 0.25 & 0 & 0.45 \end{bmatrix} \tag{2.103}$$

A very special property of markov matrices is that exactly one eigenvalue of any markov matrix equals 1 and the rest of its eigenvalues are strictly less than 0. For example, the M_3 has following eigenvalues: $1.0000, -0.2168, 0.3428, 0.0740$. The first part of this property can be proved as follows. The matrix $M - I$ is singular, because the sum of the rows is a zero vector. Therefore, $det(M - I) = 0$. Thus, $\lambda = 1$ must be an eigenvalue of M.

In probabilistic models, we often have systems of the form $\mathbf{p}_{i+1} = A\mathbf{p}_i$, $\forall\, i \geq 0$, similar to equation (2.98). A closed form solution can be obtained using the idea of (2.99)

$$\mathbf{p}_i = \sum_{i=k}^{n} \lambda_k^i c_k \mathbf{v}_k$$

where, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are the eigenvectors of M and its eigenvalues are $\lambda_1, \lambda_2, \ldots, \lambda_n$. If $\lambda_1 = 1$, then $\lambda_i < 1, \forall\ 2 \leq i \leq n$. Hence, as $i \to \infty$, $\mathbf{p}_i \to c_1\mathbf{v}_1$.

12. If A is an $n \times n$ matrix with real valued entries and is symmetric, *i.e.*, $A = A^T$, then, its eigenvalues are real. Further, the eigenvectors of a symmetric matrix can be chosen to be orthogonal. In mathematics, this is called the *spectral theeorem* while in mechanics it is called the *principal axis theorem.*

[10]The matrix entries of a markov entries represent probabilities of transitions within/between states.

Theorem 22 *If A is symmetric then (1) all its eigenvalues are real and (2) there exists and orthonormal basis Q of A, consisting of its eigenvectors.*

Proof for part (1): Let λ be an eigenvalue of A and $\mathbf{x}$ be its corresponding eigenvector; $A\mathbf{x} = \lambda\mathbf{x}$. Then, premultiplying both sides by $\overline{\mathbf{x}}^T$, we get

$$\overline{\mathbf{x}}^T A\mathbf{x} = \lambda\overline{\mathbf{x}}^T\mathbf{x} \tag{2.104}$$

As mentioned earlier, the complex conjugates $\overline{\lambda}$ and $\overline{x}$ also form an eigenvalue-eigenvector pair for a real matrix A; $A\overline{\mathbf{x}} = \overline{\lambda}\overline{\mathbf{x}}$. This implies that $\overline{\mathbf{x}}^T A^T = \overline{\mathbf{x}}^T A = \overline{\lambda}\overline{\mathbf{x}}^T$ and therefore,

$$\overline{\mathbf{x}}^T A\mathbf{x} = \overline{\lambda}\overline{\mathbf{x}}^T\mathbf{x} \tag{2.105}$$

We note that the left hand sides of (2.104) and (2.105) are the same. Equating the right hand sides of these equations,

$$\lambda\overline{\mathbf{x}}^T\mathbf{x} = \overline{\lambda}\overline{\mathbf{x}}^T\mathbf{x} \tag{2.106}$$

$\overline{\mathbf{x}}^T\mathbf{x}$ is always real and non-negative. It is 0 only if $\mathbf{x} = \mathbf{0}$. Therefore, $\lambda = \overline{\lambda} \Rightarrow \lambda \in \Re$. $\Box$

13. If A is a real symmetric matrix, the number of positive pivots and number of negative pivots are respectively equal to the number of positive and negative eigenvalues.

14. Two $n \times n$ matrices A and B are called *similar* if there exists an invertible $n \times n$ matrix M such that $M^{-1}BM = A$. A property of similar matrices is that they have same determinants, since $det(A) = det(M^{-1})det(B)det(M) = \frac{1}{det(M)}det(B)det(M) = det(B)$. A more fundamental property is that similar matrices have the same eigenvalues, though they could differ in their eigenvectors.

Theorem 23 *If A and B are similar matrices, they have the same eigenvalues.*

Proof: Let λ be an eigenvalue of A. Since A and B are similar, there exists an invertible matrix M such that, $M^{-1}BM = A$. $A\mathbf{x} = \lambda\mathbf{x} \Rightarrow (MAM^{-1})M\mathbf{x} = \lambda M\mathbf{x} \Rightarrow B(M\mathbf{x}) = \lambda(M\mathbf{x})$, that is, if λ is an eigenvalue of A and $\mathbf{x}$ is the corresponding eigenvector, then λ is an eigenvalue of B and $M\mathbf{x}$ is its corresponding eigenvector.

Similarly, $B\mathbf{x} = \lambda\mathbf{x} \Rightarrow (M^{-1}AM)M^{-1}\mathbf{x} = \lambda M^{-1}\mathbf{x} \Rightarrow B(M^{-1}\mathbf{x}) = \lambda(M^{-1}\mathbf{x})$, that is, if λ is an eigenvalue of B and $\mathbf{x}$ is the corresponding eigenvector, then λ is an eigenvalue of A and $M^{-1}\mathbf{x}$ is its corresponding eigenvector. □

At this point, we state the observation that matrices of the form $kI_{n\times n}$ are only similar to themselves, since, for any invertible matrix M, $M^{-1}(kI_{n\times n})M = kI_{n\times n}$.

2.11.3 Matrix Factorization using Eigenvectors

Let A be an $n \times n$ matrix, with n eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ and corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_1$. Let V be a matrix with the eigenvectors as columns. Postmultiplying A by V, we get

$$AV = [\lambda_1\mathbf{v}_1 \quad \lambda_2\mathbf{v}_2 \quad \ldots \quad \lambda_n\mathbf{v}_n] = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \ldots \quad \mathbf{v}_n] \underbrace{\begin{bmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ . & . & \ldots & . \\ . & . & \lambda_k & . \\ . & . & \ldots & . \\ 0 & 0 & \ldots & \lambda_n \end{bmatrix}}_{Eigenvalue\ matrix\ \Lambda} = V\Lambda$$

that is, $AV = V\Lambda$. The diagonal matrix Λ consists of eigenvalues along its diagonal and is called the *eigenvalue matrix.*

If the eigenvectors are linearly independent, V is invertible. Premultiplying AV by V^{-1},

$$V^{-1}AV = \Lambda$$

Another equivalent equation is

$$A = V\Lambda V^{-1} \tag{2.107}$$

This procedure of premultiplying a matrix by the inverse of its eigenvector matrix and post-multipyling it by the eigenvector matrix to obtain a diagonal matrix of its eigenvalues, is called *diagonalization.* Diagonalization can be generalized to powers of k:

$$A^k = V\Lambda^k V^{-1}$$

Thus, eigenvalues and eigenvectors provide a great way to understand the powers of a matrix. Further, if $|\lambda_i| < 1$, $\Lambda^k \rightarrow 0,\ as\ k \rightarrow \infty$. Therefore, if $|\lambda_i| < 1$,

$A^k \to 0,\ as\ k \to \infty$. As another example, if we define $e^{\rho A} = \sum_{n=0}^{\infty} \frac{1}{n!}(A\rho)^n$, where $\rho \in \Re$, then using the above property, it can be shown that $e^{\rho A} = Ve^{\rho\Lambda}V^{-1} = V\ diag(e^{\rho\lambda_1}, e^{\rho\lambda_2}, \ldots, e^{\rho\lambda_n})V^{-1}$, where $diag(c_1, c_2, \ldots, c_n)$ returns an $n \times n$ diagonal matrix with the i^{th} diagonal entry as c_i.

If A is symmetric, the eigenvector matrix V could be chosen to be a matrix of orthonormal vectors, denoted by Q. Note that $Q^{-1} = Q^T$. Thus, for a symmetric A, the equation (2.107) can be re-written as:

$$A = Q\Lambda Q^T = \sum_{i=1}^{n} \lambda_i(\mathbf{q}_i\mathbf{q}_i^T) \tag{2.108}$$

From Section 2.9.1, we recall that $(\mathbf{q}_i\mathbf{q}_i^T)$ is a projection matrix. Moreover, if $i \neq j$, $(\mathbf{q}_i\mathbf{q}_i^T)$ is orthogonal to $(\mathbf{q}_j\mathbf{q}_j^T)$. This gives us another perspective of symmetric matrices - as a linear combination of orthogonal projection matrices. Also, since Q is of rank 1 and invertible, we can infer that A is similar to Λ. The diagonal matrix Λ can be thought of as a canonical form for the family of matrices similar to A. However, if A is not a full rank matrix, there exists an 'almost diagonal form', called the *Jordan form* [**?**], which is similar to A, containing the eigenvalues of A along its diagonal, with the only other non-zero entries being along the super-diagonal.

One more illustration of the utility of matrix factorization using eigenvectors is the interpretation of level sets involving the quadratic form $\mathbf{x}^TA\mathbf{x} = \mathbf{x}^TQ\Lambda Q^T\mathbf{x}$ for a symmetric matrix A. The *level set* of a real-valued function f of $\mathbf{x} \in \Re^n$ is a set of the form $\{\mathbf{x}|f(\mathbf{x}) = c\}$, where c is a constant. Using the eigenvalue factorization of matrices, the level set $\{\mathbf{x}|\mathbf{x}^TQ\Lambda Q^T\mathbf{x} = c\}$ can be interpreted as an ellipsoid in n dimensions, with each eigenvector-eigenvalue pair specifying the direction and the length respectively of an axis of the ellipsoid.

2.12 Positive Definite Matrices

Positive definite matrix: *A positive definite (p.d.) matrix is a symmetric matrix with all positive eigenvalues. That M is a p.d. matrix is also denoted by $M > 0$.*

By virtue of property of symmetric matrices, all the pivots in the rref of a p.d. matrix are also positive. Since the determinant of matrix equals the product of its eigenvalues, the determinant of a p.d. matrix is also positive; however, it is not necessary that a matrix with positive determinant is also p.d.

A matrix is called *positive semi-definite* (p.s.d.), if all its eigenvalues are non-negative. That M is p.s.d. is also denoted by $M \geq 0$.

2.12.1 Equivalent Conditions

We will list down some necessary and sufficient conditions for a matrix A to be positive definite or positive semi-definite:

1. *A matrix A is p.d. iff all its principal minors (c.f. Section 2.10.1) are positive.* As an example, if A is a 2×2 matrix, we must have $a_{11} > 0$ and $a_{11}a_{22} - a_{12}a_{21} > 0$ in order for A to be p.d. On the other hand, if all its principal minors are non-negative, the matrix is p.s.d.

2. Another equivalent definition for positive definiteness is: *A matrix A is p.d. iff,* $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T A\mathbf{x} > 0$. This condition can be rewritten as $\forall\ \mathbf{x} \neq \mathbf{0},\ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\mathbf{x}_i\mathbf{x}_j > 0$. If $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T A\mathbf{x} \geq 0$, A is p.s.d.

3. The condition $\forall\ \mathbf{x} \neq \mathbf{0},\ \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}\mathbf{x}_i\mathbf{x}_j > 0$ involves a quadratic expression. The expression is guaranteed to be greater than $0\ \forall\ \mathbf{x} \neq \mathbf{0}$ *iff* it can be expressed as $\sum_{i=1}^{n}\lambda_i\left(\sum_{j=1}^{i-1}\beta_{ij}x_{ij} + x_{ii}\right)^2$, where $\lambda_i \geq 0$. This is possible *iff* A can be expressed as LDL^T, where, L is a lower traingular matrix with 1 in each diagonal entry and D is a diagonal matrix of all positive diagonal entries. Or equivalently, it should be possible to factorize A as RR^T, where $R = LD^{1/2}$ is a lower traingular matrix. Note that any symmetric matrix A can be expressed as LDL^T, where L is a lower traingular matrix with 1 in each diagonal entry and D is a diagonal matrix; positive definiteness has only an additional requirement that the diagonal entries of D be positive. This gives another equivalent condition for positive definiteness: *Matrix A is p.d. if and only if, A can be uniquely factored as $A = RR^T$, where R is a lower traingular matrix with positive diagonal entries.* This factorization of a p.d. matrix is reffered to as *Cholesky factorization.*

   Recall that Guass elimination on a matrix A yields its factorization as $A = LU$ and the diagonal entries of L are pivots. Therefore, if A is symmetric matrix such that Guass elimination on it yields positive pivots, A is positive definite.

To illustrate the equivalence of the above definitions of positive definiteness, consider the matrix P below:

$$P = \begin{bmatrix} 1 & 1 & 2 & 1 \\ 1 & 10 & 14 & 4 \\ 2 & 14 & 21 & 9 \\ 1 & 4 & 9 & 20 \end{bmatrix} \tag{2.109}$$

The matrix is positive definite and this can be proved by showing any of the following properties:

1. All the eigenvalues of P, *viz.*, $\lambda_1 = 0.1644,\ \lambda_2 = 0.9371,\ \lambda_3 = 14.4091,\ \lambda_4 = 36.4893$ are positive. and therefore $P > 0$.

2. The principal minors of P are $1, 9, 9$ and 81. All the four principal minors are positive and thus $P > 0$.

3. Matrix P can be factorized as LL^T, where

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 \\ 2 & 4 & 1 & 0 \\ 1 & 1 & 3 & 3 \end{bmatrix} \tag{2.110}$$

Since L is lower traingular and since all its diagonal entries are positive, $P > 0$.

2.12.2 Some properties

We will list some properties of positive definite matrices, using an appropriate definition of positive definiteness as required.

1. If matrices $A > 0$ and $B > 0$, then $A + B > 0$. This follows from the fact that $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T A\mathbf{x} > 0$ and $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T B\mathbf{x} > 0$ implies that $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T(A + B)\mathbf{x} > 0$. Similarly, $AB > 0$ and for any $c > 0$, $cA > 0$.

2. If $A > 0$, then $\forall\ \mathbf{x} \neq \mathbf{0}$, $\mathbf{x}^T A\mathbf{x} > 0$ implies $(\mathbf{x}^T A\mathbf{x})^T = \mathbf{x}^T A^T\mathbf{x} > 0$, that is, $A^T > 0$.

3. Let A be an $m \times n$ matrix. Recall from Section 2.9.2, the important matrix $A^T A$ which happened to be an $n \times n$ matrix. If A is full column rank, the only vector in its null space is $\mathbf{0}$. Note that $\forall\ \mathbf{x} \neq \mathbf{0},\ \mathbf{x}^T A^T A\mathbf{x} = ||A\mathbf{x}||^2 > 0$. Thus, $A^T A$ is always p.d. if A is non-singular.

4. Every p.d. matrix is invertible and its inverse is also p.d. That is, if $A > 0$ then A^{-1} exists and $A^{-1} > 0$.

5. If $A > 0$, the diagonal entries of A are real and positive. Consequently, the trace $tr(A)$ is also positive.

Testing for positive definiteness of a matrix arises in several applications, including optimization. Determining the local minimum of a function $f(\mathbf{x}),\ \mathbf{x} \in \mathcal{D},\ \mathcal{D} \subseteq \Re^k$ involves determining points $\hat{\mathbf{x}}$ at which $\nabla f(\hat{\mathbf{x}}) = 0$ and $\nabla^2 f(\hat{\mathbf{x}}) > 0$ (positive curvature at $\hat{\mathbf{x}}$).

2.13 Singular Value Decomposition

In Section 2.11.3, we discussed that a full rank symmetric matrix can be factorized into $Q\Lambda Q^T$, where, Q is an orthonormal matrix and Λ is a diagonal matrix. This factorization can be extended to any matrix and it is called *Singular Value Decomposition*, abbreviated as *SVD*. The singular value decomposition of any $m \times n$ matrix A is factorization of A as $U\Sigma V^T$, where Σ is a diagonal matrix and U and V are orthonormal matrices.

We will contruct the matrices U and V as follows. Let r be the rank of A and let

- $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r$ be an orthonormal basis for the column space of A.
- $\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_n$ be an orthonormal basis for the null space of A.
- $\mathbf{u}_{r+1}, \mathbf{u}_{r+2}, \ldots, \mathbf{u}_m$ be an orthonormal basis for the null space of A^T.
- $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ be such that $\mathbf{x}_i = A^T\mathbf{u}_i$ and $\mathbf{v}_i = \frac{1}{||\mathbf{x}_i||}\mathbf{x}_i$.

The relationship between $\mathbf{u}_i$ and $\mathbf{v}_i$ is therefore $A^T\mathbf{u}_i = \sigma_{ii}\mathbf{v}_i$, with

$$\sigma_{ii} = \begin{cases} ||A^T\mathbf{u}_i|| & \text{if } i \le r \\ 0 & \text{if } i > r \end{cases} \tag{2.111}$$

This system of equations can written in matrix form as

$$A^T U = V\Sigma \tag{2.112}$$

where, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ are the columns of U and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are the columns of V. Σ is an $n \times n$ diagonal matrix with its ij^{th} entry given by σ_{ij}, such that

$$\sigma_{ij} = \begin{cases} 0 & \text{if } i \ne j \\ ||A^T\mathbf{u}_i|| & \text{if } i = j \text{ and } i \le r \\ 0 & \text{if } i = j \text{ and } i > r \end{cases} \tag{2.113}$$

It can be shown that $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r$ are orthonormal and form a basis for the row space of A. Theorem 19 stated that the row space $C(A^T)$ and right null space $N(A)$ are orthogonal complements. Similarly, the column space $C(A)$ and left null space $N(A^T)$ are orthogonal complements. Therefore, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m$ is an orthonormal basis for $\Re^m$, while $\mathbf{v}_1, \mathbf{v}_1, \ldots, \mathbf{v}_n$ is an orthonormal basis for $\Re^n$.

Since $U^{-1} = U^T$, we can rewrite (2.112) as

$$A = U\Sigma V^T \tag{2.114}$$

Furthermore, $AA^T = U\Sigma^2 U^T$ and $A^T A = V\Sigma^2 V^T$, which are spectral decompositions, implying that the columns of U and V and eigenvectors of AA^T and $A^T A$ respectively and the diagonal entries of Σ are square roots of the eigenvalues of AA^T (or equivalently $A^T A$).

As an example, if P is the full rank, symmetric matrix in (2.109), the matrices U, Σ and V are

$$U = \begin{bmatrix} -165/2423 & 76/4167 & 637/688 & -892/2403 \\ -467/1012 & 373/992 & -577/1726 & -757/1036 \\ -367/508 & 48/133 & 318/1909 & 869/1536 \\ -172/337 & -407/477 & -329/5765 & -211/2328 \end{bmatrix} \tag{2.115}$$

$$\Sigma = \begin{bmatrix} 1715/47 & 0 & 0 & 0 \\ 0 & 11657/809 & 0 & 0 \\ 0 & 0 & 477/509 & 0 \\ 0 & 0 & 0 & 265/1612 \end{bmatrix} \tag{2.116}$$

$$V = \begin{bmatrix} -165/2423 & 76/4167 & 637/688 & -892/2403 \\ -467/1012 & 373/992 & -577/1726 & -757/1036 \\ -367/508 & 48/133 & 318/1909 & 869/1536 \\ -172/337 & -407/477 & -329/5765 & -211/2328 \end{bmatrix} \tag{2.117}$$

On the other hand, if P is a singular matrix of rank 2, given by

$$P = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 3 & 1 \\ 3 & 6 & 2 \end{bmatrix} \tag{2.118}$$

then P can be decomposed into the following matrices:

$$U = \begin{bmatrix} -1301/3398 & 794/1101 & -780/1351 \\ -450/1039 & -715/1033 & -780/1351 \\ -337/413 & 203/6999 & 780/1351 \end{bmatrix} \tag{2.119}$$

$$\Sigma = \begin{bmatrix} 2565/299 & 0 & 0 \\ 0 & 687/1076 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.120)$$

$$V = \begin{bmatrix} -799/1854 & -647/717 & 0 \\ -1814/2119 & 453/1108 & -228/721 \\ -567/1987 & 151/1108 & 684/721 \end{bmatrix} \quad (2.121)$$

Notice that, since P is singular and of rank 2, its null space has dimension 1 and one of its eigenvalues is 0.

2.13.1 Pseudoinverse

The SVD of a matrix that is not full rank (such as P in (2.118)) can be used to compute its so-called *Moore-Penrose pseudoinverse.*

Pseudoinverse: *The pseudoinverse A^+ of an $m \times n$ matrix A is a unique $n \times m$ matrix, satisfying all the following criteria:*

1. $AA^+A = A$
2. $A^+AA^+ = A^+$
3. $\overline{(AA^+)}^T = AA^+$
4. $\overline{(A^+A)}^T = A^+A$

The pseudoinverse of a non-singular square matrix is the same as its inverse. A pseudoinverse of a rectangular matrix of full column rank is the left inverse, while a pseudoinverse of a rectangular matrix of full row rank is the right inverse (*c.f.* Section 2.4.2).

Consider an $n \times n$ diagonal matrix Σ having rank k.

$$\Sigma = \begin{bmatrix} \sigma_{11} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_{22} & \dots & 0 & 0 & \dots & 0 \\ . & . & \dots & . & . & \dots & . \\ . & . & \dots & . & . & \dots & . \\ 0 & 0 & \dots & \sigma_{kk} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ . & . & \dots & . & . & \dots & . \\ . & . & \dots & . & . & \dots & . \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} \quad (2.122)$$

The pseudoinverse Σ^+ of Σ is:

$$\Sigma^+ = \begin{bmatrix} \frac{1}{\sigma_{11}} & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_{22}} & \dots & 0 & 0 & \dots & 0 \\ . & . & \dots & . & . & \dots & . \\ . & . & \dots & . & . & \dots & . \\ 0 & 0 & \dots & \frac{1}{\sigma_{kk}} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ . & . & \dots & . & . & \dots & . \\ . & . & \dots & . & . & \dots & . \\ 0 & 0 & \dots & 0 & 0 & \dots & 0 \end{bmatrix} \tag{2.123}$$

The pseudoinverse P^+ of any non full rank matrix P can be computed using its singular value decomposition $U\Sigma V^T$ and the pseudoinverse Σ^+ of the diagonal matrix Σ as:

$$P^+ = V\Sigma^+U \tag{2.124}$$

Chapter 3

Convex Optimization

3.1 Introduction

3.1.1 Mathematical Optimization

The problem of mathematical optimization is to minimize a non-linear cost function $f_0(x)$ subject to inequality constraints $f_i(x) \leq 0, i = 1, \ldots, m$ and equality constraints $h_i(x) = 0, i = 1, \ldots, p$. $x = (x_1, \ldots, x_n)$ is a vector of variables involved in the optimization problem. The general framework of a non-linear optimization problem is outlined in (3.1).

$$\begin{array}{lll} \text{minimize} & f_0(x) & \\ \text{subject to} & f_i(x) \leq 0, & i = 1, \ldots, m \\ & h_i(x) = 0, & i = 1, \ldots, p \\ \text{variable } x = (x_1, \ldots, x_n) & & \end{array} \tag{3.1}$$

It is obviously very useful and arises throughout engineering, statistics, estimation and numerical analysis. In fact there is the tautology that 'everything is an optimization problem', though the tautology does not convey anything useful. The most important thing to note first is that the optimization problem is extremely hard in general. The solution and method is very much dependent on the property of the objective function as well as properties of the functions involved in the inequality and equality constraints. There are no good methods for solving the general non-linear optimization problem. In practice, you have to make some compromises, which usually translates to finding locally optimial solutions efficiently. But then you get only suboptimial solutions, unless you are willing to do global optimizations, which is for most applications too expensive.

There are important exceptions for which the situation is much better; the global optimum in some cases can be found efficiently and relaibly. Three best known exceptions are

1. least-squares
2. linear programming
3. convex optimization problems - more or less the most general class of problems that can be solved efficiently.

Least squares and linear programming have been around for quite some time and are very special types of convex optimization problems. Convex programming was not appreciated very much until last 15 years. It has drawn attention more recently. In fact many combinatorial optimization problems have been identified to be convex optimization problems. There are also some exceptions besides convex optimization problems, such as singular value decomposition (which corresponds to the problem of finding the best rank-k approximation to a matrix, under the Frobenius norm) *etc.*, which has an exact global solution.

We will first introduce some general optimization principles. We will subsequently motivate the specific class of optimization problems called convex optimization problems and define convex sets and functions. Next, the theory of lagrange multipliers will be motivated and duality theory will be introduced. As two specific and well-studied examples of convex optimization, techniques for least squares and linear programming will be discussed to contrast them against generic convex optimization. Finally, we will dive into techniques for solving general convex optimization problems.

3.1.2 Some Topological Concepts in $\Re^n$

The definitions of some basic topological concepts in $\Re^n$ could be helpful in the discussions that follow.

Definition 22 [Balls in $\Re^n$]: *Consider a point $\mathbf{x} \in \Re^n$. Then the closed ball around $\mathbf{x}$ of radius ϵ is defined as*

$$\mathcal{B}[\mathbf{x}, \epsilon] = \{\mathbf{y} \in \Re^n | ||\mathbf{y} - \mathbf{x}|| \leq \epsilon\}$$

Likewise, the open ball around $\mathbf{x}$ of radius ϵ is defined as

$$\mathcal{B}(\mathbf{x}, \epsilon) = \{\mathbf{y} \in \Re^n | ||\mathbf{y} - \mathbf{x}|| < \epsilon\}$$

For the 1-D case, open and closed balls degenerate to open and closed intervals respectively.

Definition 23 [Boundedness in $\Re^n$]: *We say that a set $\mathcal{S} \subset \Re^n$ is bounded when there exists an $\epsilon > 0$ such that $\mathcal{S} \subseteq \mathcal{B}[0, \epsilon]$.*

In other words, a set $\mathcal{S} \subseteq \Re^n$ is bounded means that there exists a number $\epsilon > 0$ such that for all $\mathbf{x} \in \mathcal{S}$, $||\mathbf{x}|| \leq \epsilon$.

Definition 24 [Interior and Boundary points]: *A point $\mathbf{x}$ is called an interior point of a set $\mathcal{S}$ if there exists an $\epsilon > 0$ such that $\mathcal{B}(\mathbf{x}, \epsilon) \subseteq \mathcal{S}$.*

In other words, a point $\mathbf{x} \in \mathcal{S}$ is called an interior point of a set $\mathcal{S}$ if there exists an open ball of non-zero radius around $\mathbf{x}$ such that the ball is completely contained within $\mathcal{S}$.

Definition 25 [Interior of a set]: *Let $\mathcal{S} \subseteq \Re^n$. The set of all points lying in the interior of $\mathcal{S}$ is denoted by $int(\mathcal{S})$ and is called the interior of $\mathcal{S}$. That is,*

$$int(\mathcal{S}) = \{\mathbf{x} | \exists \epsilon > 0 \ s.t. \ \mathcal{B}(\mathbf{x}, \epsilon) \subset \mathcal{S}\}$$

In the 1−D case, the open interval obtained by excluding endpoints from an interval $\mathcal{I}$ is the interior of $\mathcal{I}$, denoted by $int(\mathcal{I})$. For example, $int([a, b]) = (a, b)$ and $int([0, \infty)) = (0, \infty)$.

Definition 26 [Boundary of a set]: *Let $\mathcal{S} \subseteq \Re^n$. The boundary of $\mathcal{S}$, denoted by $bnd(\mathcal{S})$ is defined as*

$$bnd(\mathcal{S}) = \left\{\mathbf{y} | \forall \ \epsilon > 0, \ \mathcal{B}(\mathbf{y}, \epsilon) \cap \mathcal{S} \neq \emptyset \ and \ \mathcal{B}(\mathbf{y}, \epsilon) \cap \mathcal{S}^C \neq \emptyset\right\}$$

For example, $bnd([a, b]) = \{a, b\}$.

Definition 27 [Open Set]: *Let $\mathcal{S} \subseteq \Re^n$. We say that $\mathcal{S}$ is an open set when, for every $\mathbf{x} \in \mathcal{S}$, there exists an $\epsilon > 0$ such that $\mathcal{B}(\mathbf{x}, \epsilon) \subset \mathcal{S}$.*

The simplest examples of an open set are the open ball, the empty set $\emptyset$ and $\Re^n$. Further, arbitrary union of opens sets is open. Also, finite intersection of open sets is open. The interior of any set is always open. It can be proved that a set $\mathcal{S}$ is open if and only if $int(\mathcal{S}) = \mathcal{S}$.

The complement of an open set is the closed set.

Definition 28 [Closed Set]: *Let $\mathcal{S} \subseteq \Re^n$. We say that $\mathcal{S}$ is a closed set when $\mathcal{S}^C$ (that is the complement of $\mathcal{S}$) is an open set.*

The closed ball, the empty set $\emptyset$ and $\Re^n$ are three simple examples of closed sets. Arbitrary intersection of closed sets is closed. Furthermore, finite union of closed sets is closed.

Definition 29 [Closure of a Set]: *Let $\mathcal{S} \subseteq \Re^n$. The closure of $\mathcal{S}$, denoted by $closure(\mathcal{S})$ is given by*

$$closure(\mathcal{S}) = \{\mathbf{y} \in \Re^n | \forall \ \epsilon > 0, \mathcal{B}(\mathbf{y}, \epsilon) \cap \mathcal{S} \neq \emptyset\}$$

Loosely speaking, the closure of a set is the smallest closed set containing the set. The closure of a closed set is the set itself. In fact, a set $\mathcal{S}$ is closed if and only if $closure(\mathcal{S}) = \mathcal{S}$. A bounded set can be defined in terms of a closed set; a set $\mathcal{S}$ is bounded if and only if it is contained inside a closed set. A relationship between the interior, boundary and closure of a set $\mathcal{S}$ is $closure(\mathcal{S}) = int(\mathcal{S}) \cup bnd(\mathcal{S})$.

3.1.3 Optimization Principles for Univariate Functions

Maximum and Minimum values of univariate functions

Let f be a function with domain $\mathcal{D}$. Then f has an *absolute maximum* (or global maximum) value at point $c \in \mathcal{D}$ if

$$f(x) \leq f(c),\ \forall x \in \mathcal{D}$$

and an *absolute minimum* (or global minimum) value at $c \in \mathcal{D}$ if

$$f(x) \geq f(c),\ \forall x \in \mathcal{D}$$

If there is an open interval $\mathcal{I}$ containing c in which $f(c) \geq f(x),\ \forall x \in \mathcal{I}$, then we say that $f(c)$ is a *local maximum value* of f. On the other hand, if there is an open interval $\mathcal{I}$ containing c in which $f(c) \leq f(x),\ \forall x \in \mathcal{I}$, then we say that $f(c)$ is a *local minimum value* of f. If $f(c)$ is either a local maximum or local minimum value of f in an open interval $\mathcal{I}$ with $c \in \mathcal{I}$, the $f(c)$ is called a *local extreme value* of f.

The following theorem gives us the first derivative test for local extreme value of f, when f is differentiable at the extremum.

Theorem 24 *If $f(c)$ is a local extreme value and if f is differentiable at $x = c$, then $f'(c) = 0$.*

Proof: Suppose $f(c) \geq f(x)$ for all x in an open interval $\mathcal{I}$ containing c and that $f'(c)$ exists. Then the difference quotient $\frac{f(c+h)-f(c)}{h} \leq 0$ for small $h \geq 0$ (so that $c + h \in \mathcal{I}$). This inequality remains true as $h \to 0$ from the right. In the limit, $f'(c) \leq 0$. Also, the difference quotient $\frac{f(c+h)-f(c)}{h} \geq 0$ for small $h \leq 0$ (so that $c+h \in \mathcal{I}$). This inequality remains true as $h \to 0$ from the left. In the limit, $f'(c) \geq 0$. Since $f'(c) \leq 0$ as well as $f'(c) \geq 0$, we must have $f'(c) = 0$[1]. $\Box$

The *extreme value theorem* is one of the most fundamental theorems in calculus concerning continuous functions on closed intervals. It can be stated as:

Theorem 25 *A continuous function $f(x)$ on a closed and bounded interval $[a, b]$ attains a minimum value $f(c)$ for some $c \in [a, b]$ and a maximum value $f(d)$ for some $d \in [a, b]$. That is, a continuous function on a closed, bounded interval attains a minimum and a maximum value.*

We must point out that either or both of the values c and d may be attained at the end points of the interval $[a, b]$. Based on theorem (24), the extreme value theorem can extended as:

Theorem 26 *A continuous function $f(x)$ on a closed and bounded interval $[a, b]$ attains a minimum value $f(c)$ for some $c \in [a, b]$ and a maximum value $f(d)$ for some $d \in [a, b]$. If $a < c < b$ and $f'(c)$ exists, then $f'(c) = 0$. If $a < d < b$ and $f'(d)$ exists, then $f'(d) = 0$.*

[1]By virtue of the *squeeze* or *sandwich theorem*

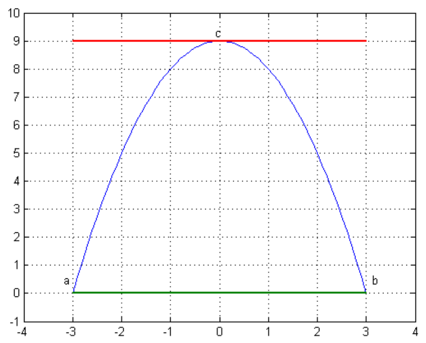


Figure 3.1: Illustration of Rolle's theorem with $f(x) = 9 - x^2$ on the interval $[-3, +3]$. We see that $f'(0) = 0$.

Next, we state the Rolle's theorem.

Theorem 27 *If f is continuous on $[a, b]$ and differentiable at all $x \in (a, b)$ and if $f(a) = f(b)$, then $f'(c) = 0$ for some $c \in (a, b)$.*

Figure 3.1 illustrates Rolle's theorem with an example function $f(x) = 9-x^2$ on the interval $[-3, +3]$.

The *mean value theorem* is a generalization of the Rolle's theorem, though we will use the Rolle's theorem to prove it.

Theorem 28 *If f is continuous on $[a, b]$ and differentiable at all $x \in (a, b)$, then there is some $c \in (a, b)$ such that, $f'(c) = \frac{f(b)-f(a)}{b-a}$.*

Proof: Define $g(x) = f(x) - \frac{f(b)-f(a)}{b-a}(x - a)$ on $[a, b]$. We note rightaway that $g(a) = g(b)$ and $g'(x) = f'(x) - \frac{f(b)-f(a)}{b-a}$. Applying Rolle's theorem on $g(x)$, we know that there exists $c \in (a, b)$ such that $g'(c) = 0$. Which implies that $f'(c) = \frac{f(b)-f(a)}{b-a}$. $\square$

Figure 3.2 illustrates the mean value theorem for $f(x) = 9 - x^2$ on the interval $[-3, 0]$. We observe that the tanget at $x = -1$ is parallel to the secant joining -3 to 0. One could think of the *mean value theorem* as a slanted version of Rolle's theorem. A natural corollary of the mean value theorem is as follows:

Corollary 29 *Let f be continuous on $[a, b]$ and differentiable on (a, b) with $m \leq f'(x) \leq M$, $\forall x \in (a, b)$. Then, $m(x - t) \leq f(x) - f(t) \leq M(x - t)$, if $a \leq t \leq x \leq b$.*

Let $\mathcal{D}$ be the domain of function f. We define

1. the linear approximation of a differentiable function $f(x)$ as $L_a(x) = f(a) + f'(a)(x - a)$ for some $a \in \mathcal{D}$. We note that $L_a(x)$ and its first derivative at a agree with $f(a)$ and $f'(a)$ respectively.

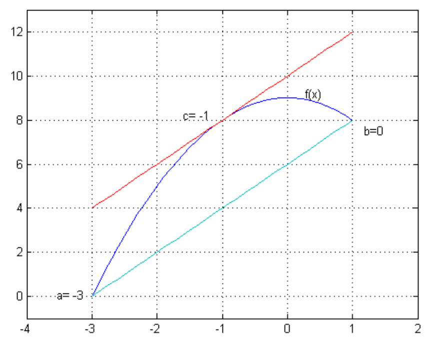


Figure 3.2: Illustration of mean value theorem with $f(x) = 9-x^2$ on the interval $[-3,0]$. We see that $f'(-1) = \frac{f(0)-f(-3)}{3}$.

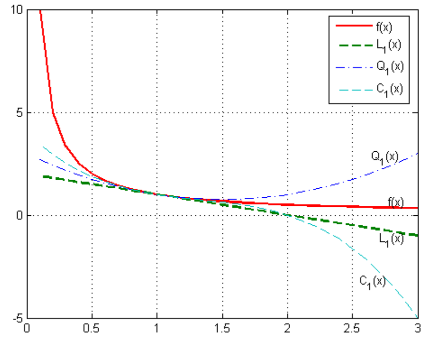


Figure 3.3: Plot of $f(x) = \frac{1}{x}$, and its linear, quadratic and cubic approximations.

2. the quadratic approximatin of a twice differentiable function $f(x)$ as the parabola $Q_a(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2$. We note that $Q_a(x)$ and its first and second derivatives at a agree with $f(a)$, $f'(a)$ and $f''(a)$ respectively.

3. the cubic approximation of a thrice differentiable function $f(x)$ is $C_a(x) = f(a)+f'(a)(x-a)+\frac{1}{2}f''(a)(x-a)^2+\frac{1}{6}f'''(a)(x-a)^3$. $C_a(x)$ and its first, second and third derivatives at a agree with $f(a)$, $f'(a)$, $f''(a)$ and $f'''(a)$ respectively.

The coefficient[2] of x^2 in $Q_a(x)$ is $\frac{1}{2}f''(a)$. Figure 3.3 illustrates the linear, quadratic and cubic approximations to the function $f(x) = \frac{1}{x}$ with $a = 1$.

[2]The parabola given by $Q_a(x)$ is strictly convex if $f''(a) > 0$ and is strictly concave if $f''(a) < 0$. Strict convexity for functions of single variable will be defined on page 168.

In general, an n^{th} degree polynomial approximation of a function can be found. Such an approximation will be used to prove a generalization of the mean value theorem, called the *Taylor's theorem.*

Theorem 30 *The Taylor's theorem states that if f and its first n derivatives $f', f'', \ldots, f^{(n)}$ are continuous on the closed interval $[a, b]$, and differentiable on (a, b), then there exists a number $c \in (a, b)$ such that*

$$f(b) = f(a)+f'(a)(b-a)+\frac{1}{2!}f''(a)(b-a)^2+\ldots+\frac{1}{n!}f^{(n)}(a)(b-a)^n+\frac{1}{(n+1)!}f^{(n+1)}(c)(b-a)^{n+1}$$

Proof: Define

$$p_n(x) = f(a) + f'(a)(x-a) + \frac{1}{2!}f''(a)(x-a)^2 + \ldots + \frac{1}{n!}f^{(n)}(a)(x-a)^n$$

and

$$\phi_n(x) = p_n(x) + \Gamma(x-a)^{n+1}$$

The polynomials $p_n(x)$ as well as $\phi_n(x)$ and their first n derivatives match f and its first n derivatives at $x = a$. We will choose a value of Γ so that

$$f(b) = p_n(b) + \Gamma(b-a)^{n+1}$$

This requires that $\Gamma = \frac{f(b)-p_n(b)}{(b-a)^{n+1}}$. Define the function $g(x) = f(x) - \phi_n(x)$ that measures the difference between function f and the approximating function $\phi_n(x)$ for each $x \in [a, b]$.

- Since $g(a) = g(b) = 0$ and since g and g' are both continuous on $[a, b]$, we can apply the Rolle's theorem to conclude that there exists $c_1 \in [a, b]$ such that $g'(c_1) = 0$.
- Similarly, since $g'(a) = g'(c_1) = 0$, and since g' and g'' are continuous on $[a, c_1]$, we can apply the Rolle's theorem to conclude that there exists $c_2 \in [a, c_1]$ such that $g''(c_2) = 0$.
- In this way, Rolle's theorem can be applied successively to $g'', g''', \ldots, g^{(n-1)}$ to imply the existence of $c_i \in (a, c_{i-1})$ such that $g^{(i)}(c_i) = 0$ for $i = 3, 4, \ldots, n+1$. Note however that $g^{(n+1)}(x) = f^{(n+1)}(x) - 0 - (n+1)!\Gamma$ which gives us the value of Γ as $\frac{f^{(n+1)}(c_{n+1})}{(n+1)!}$.

Thus,

$$f(b) = f(a)+f'(a)(b-a)+\frac{1}{2!}f''(a)(b-a)^2+\ldots+\frac{1}{n!}f^{(n)}(a)(b-a)^n+\frac{f^{(n+1)}(c_{n+1})}{(n+1)!}(x-a)^{n+1}$$

□

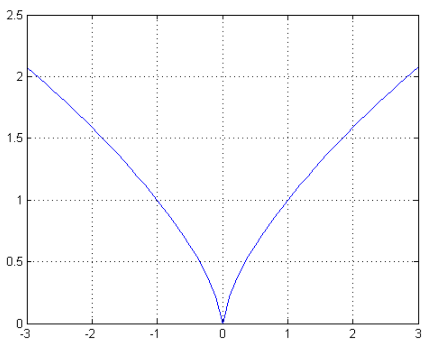


Figure 3.4: The mean value theorem can be violated if $f(x)$ is not differentiable at even a single point of the interval. Illustration on $f(x) = x^{2/3}$ with the interval $[-3, 3]$.

Note that if f fails to be differentiable at even one number in the interval, then the conclusion of the mean value theorem may be false. For example, if $f(x) = x^{2/3}$, then $f'(x) = \frac{2}{3\sqrt[3]{x}}$ and the theorem does not hold in the interval $[-3, 3]$, since f is not differentiable at 0 as can be seen in Figure 3.4.

We will introduce some definitions at this point:

- A function f is said to be *increasing* on an interval $\mathcal{I}$ in its domain $\mathcal{D}$ if $f(t) < f(x)$ whenever $t < x$.
- The function f is said to be *decreasing* on an interval $\mathcal{I} \in \mathcal{D}$ if $f(t) > f(x)$ whenever $t < x$.

These definitions help us derive the following theorem:

Theorem 31 *Let $\mathcal{I}$ be an interval and suppose f is continuous on $\mathcal{I}$ and differentiable on $int(\mathcal{I})$. Then:*

1. *if $f'(x) > 0$ for all $x \in int(\mathcal{I})$, then f is increasing on $\mathcal{I}$;*
2. *if $f'(x) < 0$ for all $x \in int(\mathcal{I})$, then f is decreasing on $\mathcal{I}$;*
3. *if $f'(x) = 0$ for all $x \in int(\mathcal{I})$, iff, f is constant on $\mathcal{I}$.*

Proof: Let $t \in \mathcal{I}$ and $x \in \mathcal{I}$ with $t < x$. By virtue of the mean value theorem, $\exists c \in (t, x)$ such that $f'(c) = \frac{f(x)-f(t)}{x-t}$.

- If $f'(x) > 0$ for all $x \in int(\mathcal{I})$, $f'(c) > 0$, which implies that $f(x) - f(t) > 0$ and we can conclude that f is increasing on $\mathcal{I}$.
- If $f'(x) < 0$ for all $x \in int(\mathcal{I})$, $f'(c) < 0$, which implies that $f(x) - f(t) < 0$ and we can conclude that f is decreasing on $\mathcal{I}$.

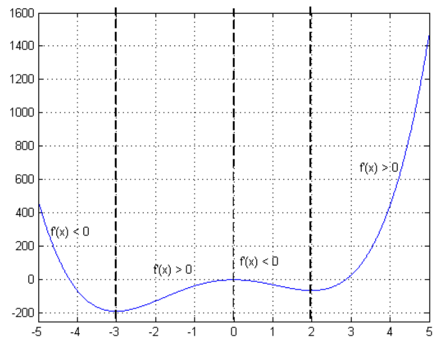


Figure 3.5: Illustration of the increasing and decreasing regions of a function $f(x) = 3x^4 + 4x^3 - 36x^2$

- If $f'(x) = 0$ for all $x \in int(\mathcal{I})$, $f'(c) = 0$, which implies that $f(x)-f(t) = 0$, and since x and t are arbitrary, we can conclude that f is constant on $\mathcal{I}$.

$\square$

Figure 3.5 illustrates the intervals in $(-\infty, \infty)$ on which the function $f(x) = 3x^4 + 4x^3 - 36x^2$ is decreasing and increasing. First we note that $f(x)$ is differentiable everywhere on $(-\infty, \infty)$ and compute $f'(x) = 12(x^3 + x^2 - 6x) = 12(x-2)(x+3)x$, which is negative in the intervals $(-\infty, -3]$ and $[0, 2]$ and positive in the intervals $[-3, 0]$ and $[2, \infty)$. We observe that f is decreasing in the intervals $(-\infty, -3]$ and $[0, 2]$ and while it is increasing in the intervals $[-3, 0]$ and $[2, \infty)$.

There is a related sufficient condition for a function f to be increasing/decreasing on an interval $\mathcal{I}$, stated through the following theorem:

Theorem 32 *Let $\mathcal{I}$ be an interval and suppose f is continuous on $\mathcal{I}$ and differentiable on $int(\mathcal{I})$. Then:*

1. *if $f'(x) \geq 0$ for all $x \in int(\mathcal{I})$, and if $f'(x) = 0$ at only finitely many $x \in \mathcal{I}$, then f is increasing on $\mathcal{I}$;*

2. *if $f'(x) \leq 0$ for all $x \in int(\mathcal{I})$, and if $f'(x) = 0$ at only finitely many $x \in \mathcal{I}$, then f is decreasing on $\mathcal{I}$.*

For example, the derivative of the function $f(x) = 6x^5 - 15x^4 + 10x^3$ vanishes at 0, and 1 and $f'(x) > 0$ elsewhere. So $f(x)$ is increasing on $(-\infty, \infty)$.

Are the sufficient conditions for increasing and decreasing properties of $f(x)$ in theorem 31 also necesssary? It turns out that it is not the case. Figure 3.6 shows that for the function $f(x) = x^5$, though $f(x)$ is increasing in $(-\infty, \infty)$, $f'(0) = 0$.

In fact, we have a slightly different necessary condition for an increasing or decreasing function.

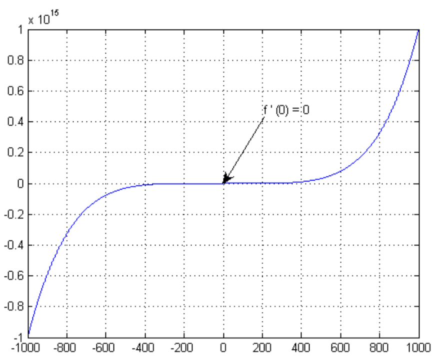


Figure 3.6: Plot of $f(x) = x^5$, illustrating that though the function is increasing on $(-\infty, \infty)$, $f'(0) = 0$.

Theorem 33 *Let $\mathcal{I}$ be an interval, and suppose f is continuous on $\mathcal{I}$ and differentiable in $int(\mathcal{I})$. Then:*

1. *if f is increasing on $\mathcal{I}$, then $f'(x) \geq 0$ for all $x \in int(\mathcal{I})$;*
2. *if f is decreasing on $\mathcal{I}$, then $f'(x) \leq 0$ for all $x \in int(\mathcal{I})$.*

Proof: Suppose f is increasing on $\mathcal{I}$, and let $x \in int(\mathcal{I})$. Them $\frac{f(x+h)-f(x)}{h} > 0$ for all h such that $x+h \in int(\mathcal{I})$. This implies that $f'(x) = \lim\limits_{h\to 0} \frac{f(x+h)-f(x)}{h} \geq 0$. For the case when f is decreasing on $\mathcal{I}$, it can be similarly proved that $f'(x) = \lim\limits_{h\to 0} \frac{f(x+h)-f(x)}{h} \leq 0$. $\Box$

Next, we define the concept of *critical number*, which will help us derive the general condition for local extrema.

Definition 30 [Critical number]: *A number c in the domain $\mathcal{D}$ of f is called a critical number of f if either $f'(c) = 0$ or $f'(c)$ does not exist.*

The general condition for local extrema is stated in the next theorem; it extends the result in theorem 24 to general non-differentiable functions.

Theorem 34 *If $f(c)$ is a local extreme value, then c is a critical number of f.*

That the converse of theorem 34 does not hold is illustrated in Figure 3.6; 0 is a critical number ($f'(0) = 0$), although $f(0)$ is not a local extreme value. Then, given a given critical number c, how do we discern whether $f(c)$ is a local extreme value? This can be answered using the *first derivative test*:

Procedure 1 [First derivative test]: *Let c be an isolated critical number of f. Then,*

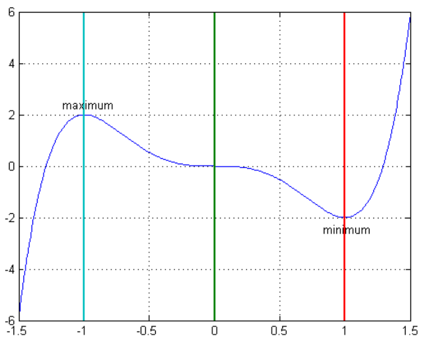


Figure 3.7: Example illustrating the derivative test for function $f(x) = 3x^5 - 5x^3$.

1. *$f(c)$ is a local minimum if $f(x)$ is decreasing in an interval $[c-\epsilon_1, c]$ and increasing in an interval $[c, c+\epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$, or equivalently, the sign of $f'(x)$ changes from negative in $[c-\epsilon_1, c]$ to positive in $[c, c+\epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$.*
2. *$f(c)$ is a local maximum if $f(x)$ is increasing in an interval $[c-\epsilon_1, c]$ and decreasing in an interval $[c, c+\epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$, or equivalently, the sign of $f'(x)$ changes from positive in $[c-\epsilon_1, c]$ to negative in $[c, c+\epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$.*
3. *If $f'(x)$ is positive in an interval $[c-\epsilon_1, c]$ and also positive in an interval $[c, c-\epsilon_2]$, or $f'(x)$ is negative in an interval $[c-\epsilon_1, c]$ and also negative in an interval $[c, c-\epsilon_2]$ with $\epsilon_1, \epsilon_2 > 0$, then $f(c)$ is not a local extremum.*

As an example, the function $f(x) = 3x^5 - 5x^3$ has the derivative $f'(x) = 15x^2(x+1)(x-1)$. The critical points are 0, 1 and -1. Of the three, the sign of $f'(x)$ changes at 1 and -1, which are local minimum and maximum respectively. The sign does not change at 0, which is therefore not a local supremum. This is pictorially depicted in Figure 3.7 As another example, consider the function

$$f(x) = \begin{cases} -x & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}$$

Then,

$$f'(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x > 0 \end{cases}$$

Note that $f(x)$ is discontinuous at $x = 0$, and therefore $f'(x)$ is not defined at $x = 0$. All numbers $x \geq 0$ are critical numbers. $f(0) = 0$ is a local minimum, whereas $f(x) = 1$ is a local minimum as well as a local maximum $\forall x > 0$.

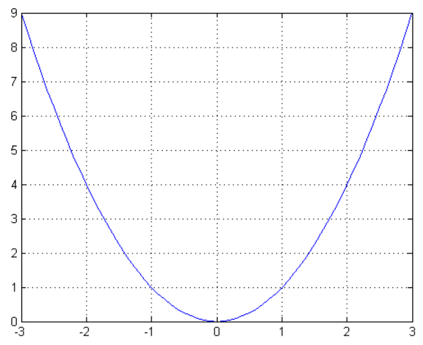


Figure 3.8: Plot for the strictly convex function $f(x) = x^2$ which has $f''(x) = 2 > 0,\ \forall x$.

Strict Convexity and Extremum

We define strictly convex and concave functions as follows:

1. A differentiable function f is said to be *strictly convex* (or *strictly concave up*) on an open interval $\mathcal{I}$, *iff*, $f'(x)$ is increasing on $\mathcal{I}$. Recall from theorem 31, the graphical interpretation of the first derivative $f'(x)$; $f'(x) > 0$ implies that $f(x)$ is increasing at x. Similarly, $f'(x)$ is increasing when $f''(x) > 0$. This gives us a sufficient condition for the strict convexity of a function:

   Theorem 35 *If at all points in an open interval $\mathcal{I}$, $f(x)$ is doubly differentiable and if $f''(x) > 0,\ \forall x \in \mathcal{I}$, then the slope of the function is always increasing with x and the graph is strictly convex. This is illustrated in Figure 3.8.*

   On the other hand, if the function is strictly convex and doubly differentiable in $\mathcal{I}$, then $f''(x) \geq 0,\ \forall x \in \mathcal{I}$.

   There is also a slopeless interpretation of strict convexity as stated in the following theorem:

   Theorem 36 *A differentiable function f is strictly convex on an open interval $\mathcal{I}$, iff*

   $$f(ax_1 + (1-a)x_2) < af(x_1) + (1-a)f(x_2) \tag{3.2}$$

   whenver $x_1, x_2 \in \mathcal{I}$, $x_1 \neq x_2$ and $0 < a < 1$.

Proof: First we will prove the necessity. Suppose f' is increasing on $\mathcal{I}$. Let $0 < a < 1$, $x_1, x_2 \in \mathcal{I}$ and $x_1 \neq x_2$. Without loss of generality assume that $x_1 < x_2$[3]. Then, $x_1 < ax_1 + (1-a)x_2 < x_2$ and therefore $ax_1 + (1-a)x_2 \in \mathcal{I}$. By the mean value theorem, there exist s and t with $x_1 < s < ax_1 + (1-a)x_2 < t < x_2$, such that $f(ax_1 + (1-a)x_2) - f(x_1) = f'(s)(x_2 - x_1)(1-a)$ and $f(x_2) - f(ax_1 + (1-a)x_2) = f'(t)(x_2 - x_1)a$. Therefore,

$$\begin{aligned} (1-a)f(x_1) - f(ax_1 + (1-a)x_2) + af(x_2) &= \\ a\left[f(x_2) - f(ax_1 + (1-a)x_2)\right] - (1-a)\left[f(ax_1 + (1-a)x_2) - f(x_1)\right] &= \\ a(1-a)(x_2 - x_1)\left[f'(t) - f'(s)\right] & \end{aligned}$$

Since $f(x)$ is strictly convex on $\mathcal{I}$, $f'(x)$ is increasing $\mathcal{I}$ and therefore, $f'(t) - f'(s) > 0$. Moreover, $x_2 - x_1 > 0$ and $0 < a < 1$. This implies that $(1-a)f(x_1) - f(ax_1 + (1-a)x_2) + af(x_2) > 0$, or equivalently, $f(ax_1 + (1-a)x_2) < af(x_1) + (1-a)f(x_2)$, which is what we wanted to prove in 3.2.

Next, we prove the sufficiency. Suppose the inequality in 3.2 holds. Therefore,

$$\lim_{a \to 0} \frac{f(x_2 + a(x_1 - x_2)) - f(x_2)}{a} \leq f(x_1) - f(x_2)$$

that is,

$$f'(x_2)(x_1 - x_2) \leq f(x_1) - f(x_2) \tag{3.3}$$

Similarly, we can show that

$$f'(x_1)(x_2 - x_1) \leq f(x_2) - f(x_1) \tag{3.4}$$

Adding the left and right hand sides of inequalities in (3.3) and (3.4), and multiplying the resultant inequality by -1 gives us

$$\left(f'(x_2) - f'(x_1)\right)(x_2 - x_1) \geq 0 \tag{3.5}$$

Using the mean value theorem, $\exists z = x_1 + t(x_2 - x_1)$ for $t \in (0,1)$ such that

[3]For the case $x_2 < x_1$, the proof is very similar.

$$f(x_2) - f(x_1) = f'(z)(x_2 - x_1) \tag{3.6}$$

Since 3.5 holds for any $x_1, x_2 \in \mathcal{I}$, it also hold for $x_2 = z$. Therefore,

$$(f'(z) - f'(x_1))(x_2 - x_1) = \frac{1}{t}(f'(z) - f'(x_1))(z - x_1) \geq 0$$

Additionally using 3.6, we get

$$f(x_2)-f(x_1) = (f'(z)-f'(x_1))(x_2-x_1)+f'(x_1)(x_2-x_1) \geq f'(x_1)(x_2-x_1) \tag{3.7}$$

Suppose equality holds in 3.5 for some $x_1 \neq x_2$. Then equality holds in 3.7 for the same x_1 and x_2. That is,

$$f(x_2) - f(x_1) = f'(x_1)(x_2 - x_1) \tag{3.8}$$

Applying 3.7 we can conclude that

$$f(x_1) + af'(x_1)(x_2 - x_1) \leq f(x_1 + a(x_2 - x_1)) \tag{3.9}$$

From 3.2 and 3.8, we can derive that

$$f(x_1 + a(x_2 - x_1)) < (1 - a)f(x_1) + af(x_2) = f(x_1) + af'(x_1)(x_2 - x_1) \tag{3.10}$$

However, equations 3.9 and 3.10 contradict each other. Therefore, equality in 3.5 cannot hold for any $x_1 \neq x_2$, implying that

$$\left(f'(x_2) - f'(x_1)\right)(x_2 - x_1) > 0$$

that is, $f'(x)$ is increasing and therefore f is convex on $\mathcal{I}$. $\square$

2. A differentiable function f is said to be *strictly concave* on an open interval $\mathcal{I}$, *iff*, $f'(x)$ is decreasing on $\mathcal{I}$. Recall from theorem 31, the graphical interpretation of the first derivative $f'(x)$; $f'(x) < 0$ implies that $f(x)$ is decreasing at x. Similarly, $f'(x)$ is monotonically decreasing when $f''(x) > 0$. This gives us a sufficient condition for the concavity of a function:

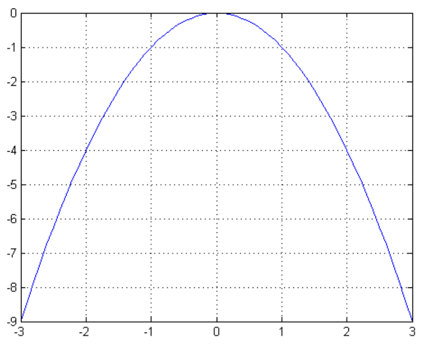


Figure 3.9: Plot for the strictly convex function $f(x) = -x^2$ which has $f''(x) = -2 < 0,\ \forall x$.

Theorem 37 *If at all points in an open interval $\mathcal{I}$, $f(x)$ is doubly differentiable and if $f''(x) < 0,\ \forall x \in \mathcal{I}$, then the slope of the function is always decreasing with x and the graph is strictly concave. This is illustrated in Figure 3.9.*

On the other hand, if the function is strictly concave and doubly differentiable in $\mathcal{I}$, then $f''(x) \leq 0,\ \forall x \in \mathcal{I}$.

There is also a slopeless interpretation of concavity as stated in the following theorem:

Theorem 38 *A differentiable function f is strictly concave on an open interval $\mathcal{I}$, iff*

$$f(ax_1 + (1-a)x_2) > af(x_1) + (1-a)f(x_2) \tag{3.11}$$

whenver $x_1, x_2 \in \mathcal{I}$, $x_1 \neq x_2$ and $0 < a < 1$.

The proof is similar to that for theorem 36.

Figure 3.10 illustrates a function $f(x) = x^3 - x + 2$, whose slope decreases as x increases to 0 ($f''(x) < 0$) and then the slope increases beyond $x = 0$ ($f''(x) > 0$). The point 0, where the $f''(x)$ changes sign is called the *inflection point*; the graph is strictly concave for $x < 0$ and strictly convex for $x > 0$. Along similar lines, we can diagnose the function $f(x) = \frac{1}{20}x^5 - \frac{7}{12}x^4 + \frac{7}{6}x^3 - \frac{15}{2}x^2$; it is strictly concave on $(-\infty, -1]$ and $[3, 5]$ and strictly convex on $[-1, 3]$ and $[5, \infty]$. The inflection points for this function are at $x = -1$, $x = 3$ and $x = 5$.

The *first derivative test* for local extrema can be restated in terms of strict convexity and concavity of functions.

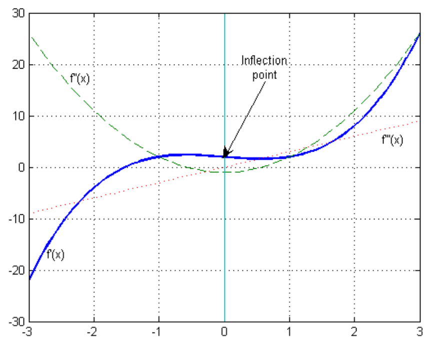


Figure 3.10: Plot for $f(x) = x^3 + x + 2$, which has an inflection point $x = 0$, along with plots for $f'(x)$ and $f''(x)$.

Procedure 2 [First derivative test in terms of strict convexity]: *Let c be a critical number of f and $f'(c) = 0$. Then,*

1. *$f(c)$ is a local minimum if the graph of $f(x)$ is strictly convex on an open interval containing c.*
2. *$f(c)$ is a local maximum if the graph of $f(x)$ is strictly concave on an open interval containing c.*

If the second derivative $f''(c)$ exists, then the strict convexity conditions for the critical number can be stated in terms of the sign of of $f''(c)$, making use of theorems 35 and 37. This is called the *second derivative test.*

Procedure 3 [Second derivative test]: *Let c be a critical number of f where $f'(c) = 0$ and $f''(c)$ exists.*

1. *If $f''(c) > 0$ then $f(c)$ is a local minimum.*
2. *If $f''(c) < 0$ then $f(c)$ is a local maximum.*
3. *If $f''(c) = 0$ then $f(c)$ could be a local maximum, a local maximum, neither or both. That is, the test fails.*

For example,

- If $f(x) = x^4$, then $f'(0) = 0$ and $f''(0) = 0$ and we can see that $f(0)$ is a local minimum.
- If $f(x) = -x^4$, then $f'(0) = 0$ and $f''(0) = 0$ and we can see that $f(0)$ is a local maximum.
- If $f(x) = x^3$, then $f'(0) = 0$ and $f''(0) = 0$ and we can see that $f(0)$ is neither a local minimum nor a local maximum. $(0, 0)$ is an inflection point in this case.

- If $f(x) = x + 2\sin x$, then $f'(x) = 1 + 2\cos x$. $f'(x) = 0$ for $x = \frac{2\pi}{3}, \frac{4\pi}{3}$, which are the critical numbers. $f''\left(\frac{2\pi}{3}\right) = 2\sin\frac{2\pi}{3} = -\sqrt{3} < 0 \Rightarrow f\left(\frac{2\pi}{3}\right) = \frac{2\pi}{3} + \sqrt{3}$ is a local maximum value. On the other hand, $f''\left(\frac{4\pi}{3}\right) = \sqrt{3} > 0$ $\Rightarrow f\left(\frac{4\pi}{3}\right) = \frac{4\pi}{3} - \sqrt{3}$ is a local minimum value.
- If $f(x) = x + \frac{1}{x}$, then $f'(x) = 1 + \frac{1}{x^2}$. The critial numbers are $x = \pm 1$. Note that $x = 0$ is not a critical number, even though $f'(0)$ does not exist, because 0 is not in the domain of f. $f''(x) = \frac{2}{x^3}$. $f''(-1) = -2 < 0$ and therefore $f(-1) = -2$ is a local maximum. $f''(1) = 2 > 0$ and therefore $f(1) = 2$ is a local minimum.

Global Extrema on Closed Intervals

Recall the extreme value theorem (theorem 25). An outcome of the extreme value theorem is that

- if either of c or d lies in (a, b), then it is a critical number of f;
- else each of c and d must lie on one of the boundaries of $[a, b]$.

This gives us a procedure for finding the maximum and minimum of a continuous function f on a closed bounded interval $\mathcal{I}$:

Procedure 4 [Finding extreme values on closed, bounded intervals]: *1. Find the critical points in int($\mathcal{I}$).*

2. Compute the values of f at the critical points and at the endpoints of the interval.

3. Select the least and greatest of the computed values.

For example, to compute the maximum and minimum values of $f(x) = 4x^3 - 8x^2 + 5x$ on the interval $[0, 1]$, we first compute $f'(x) = 12x^2 - 16x + 5$ which is 0 at $x = \frac{1}{2}, \frac{5}{6}$. Values at the critical points are $f(\frac{1}{2}) = 1$, $f(\frac{5}{6}) = \frac{25}{27}$. The values at the end points are $f(0) = 0$ and $f(1) = 1$. Therefore, the minimum value is $f(0) = 0$ and the maximum value is $f(1) = f(\frac{1}{2}) = 1$.

In this context, it is relevant to discuss the one-sided derivatives of a function at the endpoints of the closed interval on which it is defined.

Definition 31 [One-sided derivatives at endpoints]: *Let f be defined on a closed bounded interval $[a, b]$. The (right-sided) derivative of f at $x = a$ is defined as*

$$f'(a) = \lim_{h \to 0^+} \frac{f(a + h) - f(a)}{h}$$

Similarly, the (left-sided) derivative of f at $x = b$ is defined as

$$f'(b) = \lim_{h \to 0^-} \frac{f(b + h) - f(b)}{h}$$

Essentially, each of the one-sided derivatives defines one-sided slopes at the endpoints. Based on these definitions, the following result can be derived.

Theorem 39 *If f is continuous on $[a, b]$ and $f'(a)$ exists as a real number or as $\pm\infty$, then we have the following necessary conditions for extremum at a.*

- *If $f(a)$ is the maximum value of f on $[a, b]$, then $f'(a) \leq 0$ or $f'(a) = -\infty$.*
- *If $f(a)$ is the minimum value of f on $[a, b]$, then $f'(a) \geq 0$ or $f'(a) = \infty$.*

If f is continuous on $[a, b]$ and $f'(b)$ exists as a real number or as $\pm\infty$, then we have the following necessary conditions for extremum at b.

- *If $f(b)$ is the maximum value of f on $[a, b]$, then $f'(b) \geq 0$ or $f'(b) = \infty$.*
- *If $f(b)$ is the minimum value of f on $[a, b]$, then $f'(b) \leq 0$ or $f'(b) = -\infty$.*

The following theorem gives a useful procedure for finding extrema on closed intervals.

Theorem 40 *If f is continuous on $[a, b]$ and $f''(x)$ exists for all $x \in (a, b)$. Then,*

- *If $f''(x) \leq 0$, $\forall x \in (a, b)$, then the minimum value of f on $[a, b]$ is either $f(a)$ or $f(b)$. If, in addition, f has a critical number $c \in (a, b)$, then $f(c)$ is the maximum value of f on $[a, b]$.*
- *If $f''(x) \geq 0$, $\forall x \in (a, b)$, then the maximum value of f on $[a, b]$ is either $f(a)$ or $f(b)$. If, in addition, f has a critical number $c \in (a, b)$, then $f(c)$ is the minimum value of f on $[a, b]$.*

The next theorem is very useful for finding global extrema values on open intervals.

Theorem 41 *Let $\mathcal{I}$ be an open interval and let $f''(x)$ exist $\forall x \in \mathcal{I}$.*

- *If $f''(x) \geq 0$, $\forall x \in \mathcal{I}$, and if there is a number $c \in \mathcal{I}$ where $f'(c) = 0$, then $f(c)$ is the global minimum value of f on $\mathcal{I}$.*
- *If $f''(x) \leq 0$, $\forall x \in \mathcal{I}$, and if there is a number $c \in \mathcal{I}$ where $f'(c) = 0$, then $f(c)$ is the global maximum value of f on $\mathcal{I}$.*

For example, let $f(x) = \frac{2}{3}x - \sec x$ and $\mathcal{I} = (\frac{-\pi}{2}, \frac{\pi}{2})$. $f'(x) = \frac{2}{3} - \sec x \tan x = \frac{2}{3} - \frac{\sin x}{\cos^2 x} = 0 \Rightarrow x = \frac{\pi}{6}$. Further, $f''(x) = -\sec x(\tan^2 x + \sec^2 x) < 0$ on $(\frac{-\pi}{2}, \frac{\pi}{2})$. Therefore, f attains the maximum value $f(\frac{\pi}{6}) = \frac{\pi}{9} - \frac{2}{\sqrt{3}}$ on $\mathcal{I}$.

As another example, let us find the dimensions of the cone with minimum volume that can contain a sphere with radius R. Let h be the height of the cone and r the radius of its base. The objective to be minimized is the volume $f(r, h) = \frac{1}{3}\pi r^2 h$. The constraint betwen r and h is shown in Figure 3.11; the traingle AEF is similar to traingle ADB and therefore, $\frac{h-R}{R} = \frac{\sqrt{h^2+r^2}}{r}$. Our

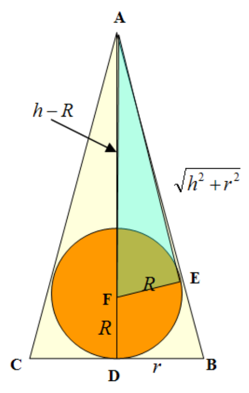


Figure 3.11: Illustrating the constraints for the optimization problem of finding the cone with minimum volume that can contain a sphere of radius R.

first step is to reduce the volume formula to involve only one of r^2[4] or h. The algebra involved will be the simplest if we solved for h. The constraint gives us $r^2 = \frac{R^2h}{h-2R}$. Substituting this expression for r^2 into the volume formula, we get $g(h) = \frac{\pi R^2}{3}\frac{h^2}{(h-2R)}$ with the domain given by $\mathcal{D} = \{h|2R < h < \infty\}$. Note that $\mathcal{D}$ is an open interval. $g' = \frac{\pi R^2}{3}\frac{2h(h-2R)-h^2}{(h-2R)^2} = \frac{\pi R^2}{3}\frac{h(h-4R)}{(h-2R)^2}$ which is 0 in its domain $\mathcal{D}$ if and only if $h = 4R$. $g'' = \frac{\pi R^2}{3}\frac{2(h-2R)^3-2h(h-4R)(h-2R)^2}{(h-2R)^4} = \frac{\pi R^2}{3}\frac{2(h^2-4Rh+4R^2-h^2+4Rh)}{(h-2R)^3} = \frac{\pi R^2}{3}\frac{8R^2}{(h-2R)^3}$, which is greater than 0 in $\mathcal{D}$. Therefore, g (and consequently f) has a unique minimum at $h = 4R$ and correspondingly, $r^2 = \frac{R^2h}{h-2R} = 2R^2$.

3.1.4 Optimization Principles for Multivariate Functions

Directional derivative and the gradient vector

Consider a function $f(\mathbf{x})$, with $\mathbf{x} \in \Re^n$. We start with the concept of the the direction at a point $\mathbf{x} \in \Re^n$. We will represent a vector by $\mathbf{x}$ and the k^{th} component of $\mathbf{x}$ by x_k. Let $\mathbf{u}^k$ be a unit vector pointing along the k^{th} coordinate axis in $\Re^n$; $u^k_k = 1$ and $u^k_j = 0$, $\forall j \neq k$ An arbitrary direction vector $\mathbf{v}$ at $\mathbf{x}$ is a vector in $\Re^n$ with unit norm (*i.e.*, $||\mathbf{v}|| = 1$) and component v_k in the direction of $\mathbf{u}^k$. Let $f : \mathcal{D} \rightarrow \Re$, $\mathcal{D} \subseteq \Re^n$ be a function.

Definition 32 [Directional derivative]: *The directional derivative of* $f(\mathbf{x})$ *at* $\mathbf{x}$ *in the direction of the unit vector* $\mathbf{v}$ *is*

$$D_{\mathbf{v}}f(\mathbf{x}) = \lim_{h\rightarrow 0}\frac{f(\mathbf{x}+h\mathbf{v})-f(\mathbf{x})}{h} \tag{3.12}$$

[4]Since r appears in the volume formula only in terms of r^2.

provided the limit exists.

As a special case, when $\mathbf{v} = \mathbf{u}^k$ the directional derivative reduces to the partial derivative of f with respect to x_k.

$$D_{\mathbf{u}^k} f(\mathbf{x}) = \frac{\partial f(\mathbf{x})}{\partial x_k}$$

Theorem 42 *If $f(\mathbf{x})$ is a differentiable function of $\mathbf{x} \in \Re^n$, then f has a directional derivative in the direction of any unit vector $\mathbf{v}$, and*

$$D_{\mathbf{v}} f(\mathbf{x}) = \sum_{k=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_k} v_k \tag{3.13}$$

Proof: Define $g(h) = f(\mathbf{x} + \mathbf{v}h)$. Now:

- $g'(0) = \lim\limits_{h\to 0} \frac{g(0+h)-g(0)}{h} = \lim\limits_{h\to 0} \frac{f(\mathbf{x}+h\mathbf{v})-f(\mathbf{x})}{h}$, which is the expression for the directional derivative defined in equation 3.12. Thus, $g'(0) = D_{\mathbf{v}} f(\mathbf{x})$.
- By definition of the chain rule for partial differentiation, we get another expression for $g'(0)$; $g'(0) = \displaystyle\sum_{k=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_k} v_k$

Therefore, $g'(0) = D_{\mathbf{v}} f(\mathbf{x}) = \displaystyle\sum_{k=1}^{n} \frac{\partial f(\mathbf{x})}{\partial x_k} v_k$ $\Box$

The theorem works if the function is differentiable at the point, else it is not predictable. The above theorem leads us directly to the idea of the gradient. We can see that the right hand side of (3.13) can be realized as the dot product of two vectors, *viz.*, $\left[\frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right]^T$ and $\mathbf{v}$. Let us denote $\frac{\partial f(\mathbf{x})}{\partial x_i}$ by $f_{x_i}(\mathbf{x})$. Then we assign a name to the special vector discovered above.

Definition 33 [Gradient Vector]: *If f is differentiable function of $\mathbf{x} \in \Re^n$, then the gradient of $f(\mathbf{x})$ is the vector function $\nabla f(\mathbf{x})$, defined as:*

$$\nabla f(\mathbf{x}) = [f_{x_1}(\mathbf{x}),\ f_{x_2}(\mathbf{x}), \ldots,\ f_{x_n}(\mathbf{x})]$$

The directional derivative of a function f at a point $\mathbf{x}$ in the direction of a unit vector $\mathbf{v}$ can be now written as

$$D_{\mathbf{v}} f(\mathbf{x}) = \nabla^T f(\mathbf{x}).\mathbf{v} \tag{3.14}$$

What does the gradient $\nabla f(\mathbf{x})$ tell you about the function $f(\mathbf{x})$? We will illustrate with some examples. Consider the polynomial $f(x,y,z) = x^2y + z\sin xy$ and the unit vector $\mathbf{v}^T = \frac{1}{\sqrt{3}}[1,1,1]^T$. Consider the point $p_0 = (0,1,3)$. We will compute the directional derivative of f at p_0 in the direction of $\mathbf{v}$. To do this, we first compute the gradient of f in general: $\nabla f = \left[2xy + yz\cos xy,\ x^2 + xz\cos xy,\ \sin xy\right]^T$. Evaluating the gradient at a specific point p_0, $\nabla f(0,1,3) = [3,\ 0,\ 0]^T$. The directional derivative at p_0 in the direction $\mathbf{v}$ is $D_{\mathbf{v}}f(0,1,3) = [3,0,0].\frac{1}{\sqrt{3}}[1,1,1]^T = \sqrt{3}$. This directional derivative is the rate of change of f at p_0 in the direction $\mathbf{v}$; it is positive indicating that the function f increases at p_0 in the direction $\mathbf{v}$. All our ideas about first and second derivative in the case of a single variable carry over to the directional derivative.

As another example, let us find the rate of change of $f(x,y,z) = e^{xyz}$ at $p_0 = (1,2,3)$ in the direction from $p_1 = (1,2,3)$ to $p_2 = (-4,6,-1)$. We first construct a unit vector from p_1 to p_2; $\mathbf{v} = \frac{1}{\sqrt{57}}[-5,4,-4]$. The gradient of f in general is $\nabla f = [yze^{xyz},\ xze^{xyz},\ xye^{xyz}] = e^{xyz}[yz,\ xz,\ xy]$. Evaluating the gradient at a specific point p_0, $\nabla f(1,2,3) = e^6\,[6,3,2]^T$. The directional derivative at p_0 in the direction $\mathbf{v}$ is $D_{\mathbf{u}}f(1,2,3) = e^6[6,3,2].\frac{1}{\sqrt{57}}[-5,4,-4]^T = e^6\frac{-26}{\sqrt{57}}$. This directional derivative is negative, indicating that the function f decreases at p_0 in the direction from p_1 to p_2.

While there exist infinitely many direction vectors $\mathbf{v}$ at any point $\mathbf{x}$, there is a unique gradient vector $\nabla f(\mathbf{x})$. Since we seperated $D_{\mathbf{v}}f(\mathbf{x})$ as the dot prouduct of $\nabla f(\mathbf{x})$ with $\mathbf{v}$, we can study $\nabla f(\mathbf{x})$ independently. What does the gradient vector tell us? We will state a theorem to answer this question.

Theorem 43 *Suppose f is a differentiable function of $\mathbf{x} \in \Re^n$. The maximum value of the directional derivative $D_{\mathbf{v}}f(\mathbf{x})$ is $||\nabla f(\mathbf{x}||$ and it is so when $\mathbf{v}$ has the same direction as the gradient vector $\nabla f(\mathbf{x})$.*

Proof: The *cauchy schwartz inequality* when applied in the eucledian space states that $|\mathbf{x}^T.\mathbf{y}| \le ||\mathbf{x}||.||\mathbf{y}||$ for any $\mathbf{x},\mathbf{y} \in \Re^n$, with equality holding *iff* $\mathbf{x}$ and $\mathbf{y}$ are linearly dependent. The inequality gives upper and lower bounds on the dot product between two vectors; $-||\mathbf{x}||.||\mathbf{y}|| \le \mathbf{x}^T.\mathbf{y} \le ||\mathbf{x}||.||\mathbf{y}||$. Applying these bounds to the right hand side of 3.14 and using the fact that $||\mathbf{v}|| = 1$, we get

$$-||\nabla f(\mathbf{x})|| \le D_{\mathbf{v}}f(\mathbf{x}) = \nabla^T f(\mathbf{x}).\mathbf{v} \le ||\nabla f(\mathbf{x})||$$

with equality holding *iff* $\mathbf{v} = k\nabla f(\mathbf{x})$ for some $k \ge 0$. Since $||\mathbf{v}|| = 1$, equality can hold *iff* $\mathbf{v} = \frac{\nabla f(\mathbf{x})}{||\nabla f(\mathbf{x})||}$. $\square$

The theorem implies that the maximum rate of change of f at a point $\mathbf{x}$ is given by the norm of the gradient vector at $\mathbf{x}$. And the direction in which the rate of change of f is maximum is given by the unit vector $\frac{\nabla f(\mathbf{x}}{||\nabla f(\mathbf{x}||}$.

An associated fact is that the minimum value of the directional derivative $D_{\mathbf{v}}f(\mathbf{x})$ is $-||\nabla f(\mathbf{x}||$ and it occurs when $\mathbf{v}$ has the opposite direction of the gradient vector, *i.e.*, $-\frac{\nabla f(\mathbf{x}}{||\nabla f(\mathbf{x}||}$. This fact is often used in numerical analysis when one is trying to minimize the value of very complex functions. The method

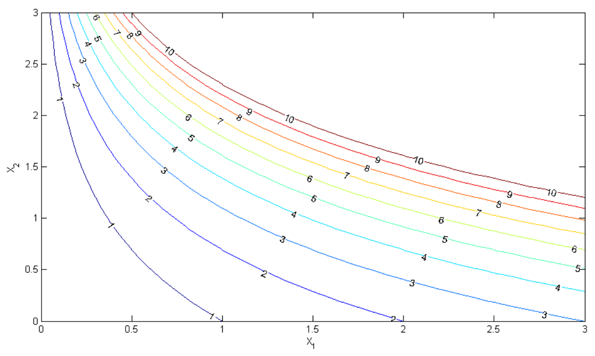


Figure 3.12: 10 level curves for the function $f(x_1, x_2) = x_1 e^{x_2}$.

of steepest descent uses this result to iteratively choose a new value of $\mathbf{x}$ by traversing in the direction of $-\nabla f(\mathbf{x})$.

Consider the function $f(x_1, x_2) = x_1 e^{x_2}$. Figure 3.12 shows 10 level curves for this function, corresponding to $f(x_1, x_2) = c$ for $c = 1, 2, \ldots, 10$. The idea behind a level curve is that as you change $\mathbf{x}$ along any level curve, the function value remains unchanged, but as you move $\mathbf{x}$ across level curves, the function value changes.

We will define the concept of a hyperplane next, since it will be repeatedly referred to in the sequel.

Definition 34 [Hyperplane]: *A set of points $\mathcal{H} \subseteq \Re^n$ is called a hyperplane if there exists a vector $\mathbf{v} \in \Re^n$ and a point $\mathbf{q} \in \Re^n$ such that*

$$\forall\ \mathbf{p} \in \mathcal{H}, (\mathbf{p} - \mathbf{q})^T \mathbf{v} = 0$$

or in other words, $\forall \mathbf{p} \in \mathcal{H}, \mathbf{p}^T\mathbf{v} = \mathbf{q}^T\mathbf{v}$. This is the equation of a hyperplane orthogonal to vector $\mathbf{v}$ and passing through point $\mathbf{q}$. The space spanned by vectors in the hyperplane $\mathcal{H}$ which are orthogonal to vector v, forms the orthogonal complement of the space spanned by v.

Hyperplane $\mathcal{H}$ can also be equivalently defined as the set of points $\mathbf{p}$ such that $\mathbf{p}^T\mathbf{v} = c$ for some $c \in \Re$ and some $\mathbf{v} \in \Re^n$, with $c = \mathbf{q}^T\mathbf{v}$ in our definition. (This definition will be referred to at a later point.)

What if $D_{\mathbf{v}} f(\mathbf{x})$ turns out to be 0? What can we say about $\nabla f(\mathbf{x})$ and $\mathbf{v}$? There is a useful theorem in this regard.

Theorem 44 *Let $f : \mathcal{D} \rightarrow \Re$ with $\mathcal{D} \in \Re^n$ be a differentiable function. The gradient ∇f evaluated at $\mathbf{x}^*$ is orthogonal to the tangent hyperplane (tangent line in case $n = 2$) to the level surface of f passing through $\mathbf{x}^*$.*

Proof: Let $\mathcal{K}$ be the range of f and let $k \in \mathcal{K}$ such that $f(\mathbf{x}^*) = k$. Consider the level surface $f(\mathbf{x}) = k$. Let $\mathbf{r}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]$ be a curve on the level surface, parametrized by $t \in \Re$, with $\mathbf{r}(0) = \mathbf{x}^*$. Then, $f(x(t), y(t), z(t)) = k$. Applying the chain rule

$$\frac{df(\mathbf{r}(t))}{dt} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\frac{dx_i(t)}{dt} = \nabla^T f(\mathbf{x}(t))\frac{d\mathbf{r}(t)}{dt} = 0$$

For $t = 0$, the equations become

$$\nabla^T f(\mathbf{x}^*)\frac{d\mathbf{r}(0)}{dt} = 0$$

Now, $\frac{d\mathbf{r}(t)}{dt}$ represents any tangent vector to the curve through $\mathbf{r}(t)$ which lies completely on the level surface. That is, the tangent line to any curve at $\mathbf{x}^*$ on the level surface containing $\mathbf{x}^*$, is orthogonal to $\nabla f(\mathbf{x}^*)$. Since the tangent hyperplane to a surface at any point is the hyperplane containing all tangent vectors to curves on the surface passing through the point, the gradient is perpendicular to the tangent hyperplane to the level surface passing through that point. The equation of the tangent hyperplane is given by $(\mathbf{x}-\mathbf{x}^*)^T\nabla f(\mathbf{x}^*) = 0$. $\square$

Recall from elementary calculus, that the normal to a plane can be found by taking the cross product of any two vectors lying within the plane. The gradient vector at any point on the level surface of a function is normal to the tangent hyperplane (or tangent line in the case of two variables) to the surface at the same point, but can however be conveniently obtained using the partial derivatives of the function at that point.

We will use some illustrative examples to study these facts.

1. Consider the same plot as in Figure 3.12 with a gradient vector at $(2, 0)$ as shown in Figure 3.13. The gradient vector $[1,\ 2]^T$ is perpendicular to the tangent hyperplane to the level curve $x_1e^{x_2} = 2$ at $(2, 0)$. The equation of the tangent hyperplane is $(x_1 - 2) + 2(x_2 - 0) = 0$ and it turns out to be a tangent line.

2. The level surfaces for $f(x_1, x_2, x_3) = x_1^2+x_2^2+x_3^2$ are shown in Figure 3.14. The gradient at $(1, 1, 1)$ is orthogonal to the tangent hyperplane to the level surface $f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 = 3$ at $(1, 1, 1)$. The gradient vector at $(1, 1, 1)$ is $[2,\ 2,\ 2]^T$ and the tanget hyperplane has the equation $2(x_1-1)+2(x_2-1)+2(x_3-1) = 0$, which is a plane in $3D$. On the other hand, the dotted line in Figure 3.15 is not orthogonal to the level surface, since it does not coincide with the gradient.

3. Let $f(x_1, x_, x_3) = x_1^2x_2^3x_3^4$ and consider the point $\mathbf{x}^0 = (1, 2, 1)$. We will find the equation of the tangent plane to the level surface through $\mathbf{x}^0$. The level surface through $\mathbf{x}^0$ is determined by setting f equal to its value evaluated at $\mathbf{x}^0$; that is, the level surface will have the equation $x_1^2x_2^3x_3^4 = 1^22^31^4 = 8$. The gradient vector (normal to tangent plane) at

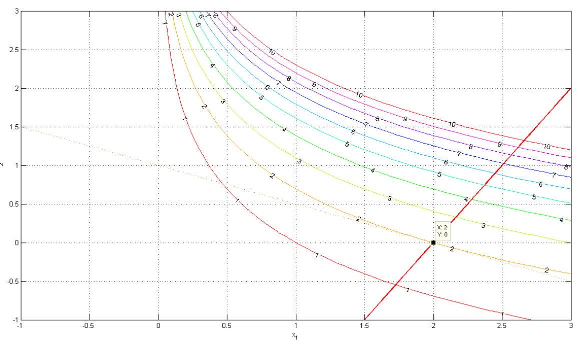


Figure 3.13: The level curves from Figure 3.12 along with the gradient vector at $(2, 0)$. Note that the gradient vector is perpendicular to the level curve $x_1 e^{x_2} = 2$ at $(2, 0)$.

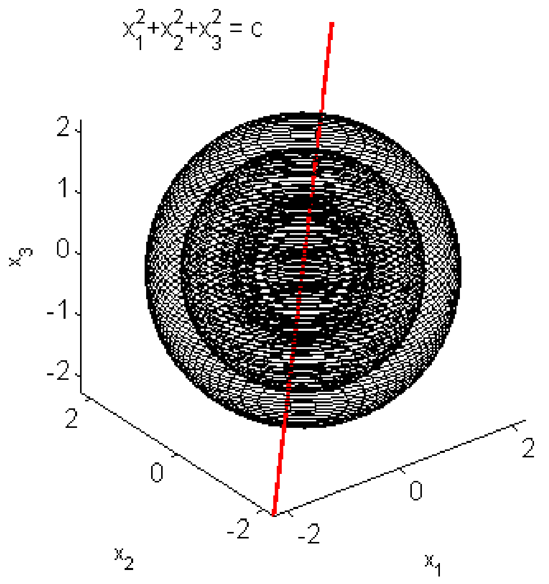


Figure 3.14: 3 level surfaces for the function $f(x_1, x_2, x_3) = x_1^2+x_2^2+x_3^2$ with $c = 1, 3, 5$. The gradient at $(1, 1, 1)$ is orthogonal to the level surface $f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 = 3$ at $(1, 1, 1)$.

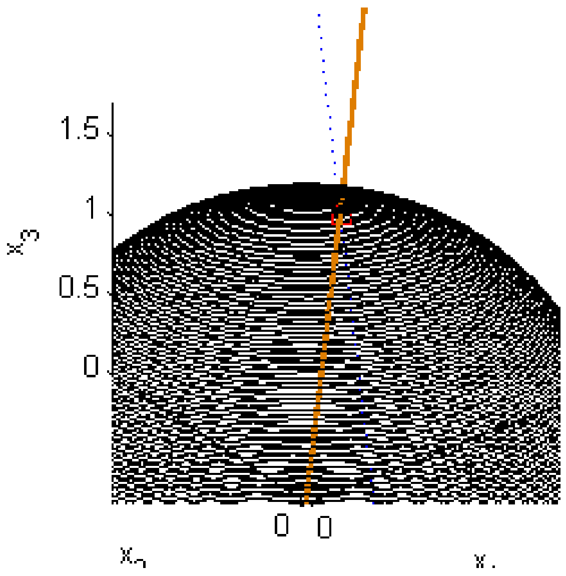


Figure 3.15: Level surface $f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^2 = 3$. The gradient at $(1, 1, 1)$, drawn as a bold line, is perpendicular to the tangent plane to the level surface at $(1, 1, 1)$, whereas, the dotted line, though passing through $(1, 1, 1)$ is not perpendicular to the same tangent plane.

$(1,2,1)$ is $\nabla f(x_1,x_2,x_3)|_{(1,2,1)} = [2x_1x_2^3x_3^4,\ 3x_1^2x_2^2x_3^4, 4x_1^2x_2^3x_3^3]^T\big|_{(1,2,1)} = [16,\ 12,\ 32]^T$. The equation of the tangent plane at $\mathbf{x}^0$, given the normal vector $\nabla f(\mathbf{x}^0)$ can be easily written down: $\nabla f(\mathbf{x}^0)^T.[\mathbf{x}-\mathbf{x}^0] = 0$ which turns out to be $16(x_1-1)+12(x_2-2)+32(x_3-1) = 0$, a plane in $3D$.

4. Consider the function $f(x,y,z) = \frac{x}{y+z}$. The directional derivative of f in the direction of the vector $\mathbf{v} = \frac{1}{\sqrt{14}}[1,\ 2,\ 3]$ at the point $x^0 = (4,1,1)$ is $\nabla^T f\big|_{(4,1,1)} . \frac{1}{\sqrt{14}}[1,\ 2,\ 3]^T = \left[\frac{1}{y+z},\ -\frac{x}{(y+z)^2},\ -\frac{x}{(y+z)^2}\right]\Big|_{(4,1,1)} . \frac{1}{\sqrt{14}}[1,\ 2,\ 3]^T = \left[\frac{1}{2},\ -1,\ -1\right] . \frac{1}{\sqrt{14}}[1,\ 2,\ 3]^T = -\frac{9}{2\sqrt{14}}$. The directional derivative is negative, indicating that the function decreases along the direction of $\mathbf{v}$. Based on theorem 43, we know that the maximum rate of change of a function at a point $\mathbf{x}$ is given by $||\nabla f(\mathbf{x})||$ and it is in the direction $\frac{\nabla f(\mathbf{x})}{||\nabla f(\mathbf{x})||}$. In the example under consideration, this maximum rate of change at $\mathbf{x}^0$ is $\frac{3}{2}$ and it is in the direction of the vector $\frac{2}{3}\left[\frac{1}{2},\ -1,\ -1\right]$.

5. Let us find the maximum rate of change of the function $f(x,y,z) = x^2y^3z^4$ at the point $\mathbf{x}^0 = (1,1,1)$ and the direction in which it occurs. The gradient at $\mathbf{x}^0$ is $\nabla^T f\big|_{(1,1,1)} = [2,\ 3,\ 4]$. The maximum rate of change at $\mathbf{x}^0$ is therefore $\sqrt{29}$ and the direction of the corresponding rate of change is $\frac{1}{\sqrt{29}}[2,\ 3,\ 4]$. The minimum rate of change is $-\sqrt{29}$ and the corresponding direction is $-\frac{1}{\sqrt{29}}[2,\ 3,\ 4]$.

6. Let us determine the equations of (a) the tangent plane to the paraboloid $\mathcal{P} : x_1 = x_2^2 + x_3^2 + 2$ at $(-1,1,0)$ and (b) the normal line to the tangent plane. To realize this as the level surface of a function of three variables, we define the function $f(x_1,x_2,x_3) = x_1 - x_2^2 - x_3^2$ and find that the paraboloid $\mathcal{P}$ is the same as the level surface $f(x_1,x_2,x_3) = -2$. The normal to the tangent plane to $\mathcal{P}$ at $\mathbf{x}^0$ is in the direction of the gradient vector $\nabla f(\mathbf{x}^0) = [1,-2,0]^T$ and its parametric equation is $[x_1,\ x_2,\ x_3] = [-1+t,\ 1-2t,\ 0]$. The equation of the tangent plane is therefore $(x_1+1) - 2(x_2-1) = 0$.

We can embed the graph of a function of n variables as the 0-level surface of a function of $n+1$ variables. More concretely, if $f : \mathcal{D} \to \Re,\ \mathcal{D} \subseteq \Re^n$ then we define $F : \mathcal{D}' \to \Re,\ \mathcal{D}' = \mathcal{D} \times \Re$ as $F(\mathbf{x},z) = f(\mathbf{x}) - z$ with $\mathbf{x} \in \mathcal{D}'$. The function f then corresponds to a single level surface of F given by $F(\mathbf{x},z) = 0$. In other words, the $0-$level surface of F gives back the graph of f. The gradient of F at any point $(\mathbf{x},z)$ is simply, $\nabla F(\mathbf{x},z) = [f_{x_1}, f_{x_2}, \ldots, f_{x_n}, -1]$ with the first n components of $\nabla F(\mathbf{x},z)$ given by the n components of $\nabla f(\mathbf{x})$. We note that the level surface of F passing through point $(\mathbf{x}^0, f(\mathbf{x}^0)$ is its 0-level surface, which is essentially the surface of the function $f(\mathbf{x})$. The equation of the tangent hyperplane to the $0-$level surface of F at the point $(\mathbf{x}^0, f(\mathbf{x}^0)$ (that is, the tangent hyperplane to $f(\mathbf{x})$ at the point $\mathbf{x}_0$), is $\nabla F(\mathbf{x}^0, f(\mathbf{x}^0))^T.[\mathbf{x}-\mathbf{x}^0, z - f(\mathbf{x}^0)]^T = 0$. Substituting appropriate expression for $\nabla F(\mathbf{x}^0)$, the equation of the tangent plane can be written as

$$\left(\sum_{i=1}^{n} f_{x_i}(\mathbf{x}^0)(x_i - x_i^0)\right) - \big(z - f(\mathbf{x}^0)\big) = 0$$

or equivalently as,

$$\left(\sum_{i=1}^{n} f_{x_i}(\mathbf{x}^0)(x_i - x_i^0)\right) + f(\mathbf{x}^0) = z$$

As an example, consider the paraboloid, $f(x_1, x_2) = 9 - x_1^2 - x_2^2$, the corresponding $F(x_1, x_2, z) = 9 - x_1^2 - x_2^2 - z$ and the point $x^0 = (\mathbf{x}^0, z) = (1, 1, 7)$ which lies on the 0-level surface of F. The gradient $\nabla F(x_1, x_2, z)$ is $[-2x_1,\ -2x_2,\ -1]$, which when evaluated at $x^0 = (1, 1, 7)$ is $[-2,\ -2,\ -1]$. The equation of the tangent plane to f at x^0 is therefore given by $-2(x_1 - 1) - 2(x_2 - 1) + 7 = z$.

Recall from theorem 24 that for functions of single variable, at local extreme points, the tangent to the curve is a line with a constant component in the direction of the function and is therefore parallel to the x-axis. If the function is is differentiable at the extreme point, then the derivative must vanish. This idea can be extended to functions of multiple variables. The requirement in this case turns out to be that the tangent plane to the function at any extreme point must be parallel to the plane $z = 0$. This can happen if and only if the gradient ∇F is parallel to the $z-$axis at the extreme point, or equivalently, the gradient to the function f must be the zero vector at every extreme point.

We will formalize this discussion by first providing the definitions for local maximum and minimum as well as absolute maximum and minimum values of a function of n variables.

Definition 35 [Local maximum]: *A function f of n variables has a local maximum at $\mathbf{x}^0$ if $\exists \epsilon > 0$ such that $\forall\ ||\mathbf{x} - \mathbf{x}^0|| < \epsilon$. $f(\mathbf{x}) \leq f(\mathbf{x}^0)$. In other words, $f(\mathbf{x}) \leq f(\mathbf{x}^0)$ whenever $\mathbf{x}$ lies in some circular disk around $\mathbf{x}^0$.*

Definition 36 [Local minimum]: *A function f of n variables has a local minimum at $\mathbf{x}^0$ if $\exists \epsilon > 0$ such that $\forall\ ||\mathbf{x} - \mathbf{x}^0|| < \epsilon$. $f(\mathbf{x}) \geq f(\mathbf{x}^0)$. In other words, $f(\mathbf{x}) \geq f(\mathbf{x}^0)$ whenever $\mathbf{x}$ lies in some circular disk around $\mathbf{x}^0$.*

These definitions are exactly analogous to the definitions for a function of single variable. Figure 3.16 shows the plot of $f(x_1, x_2) = 3x_1^2 - x_1^3 - 2x_2^2 + x_2^4$. As can be seen in the plot, the function has several local maxima and minima.

We will next state a theorem fundamental to determining the locally extreme values of functions of multiple variables.

Theorem 45 *If $f(\mathbf{x})$ defined on a domain $\mathcal{D} \subseteq \Re^n$ has a local maximum or minimum at $\mathbf{x}^*$ and if the first-order partial derivatives exist at $\mathbf{x}^*$, then $f_{x_i}(\mathbf{x}^*) = 0$ for all $1 \leq i \leq n$.*

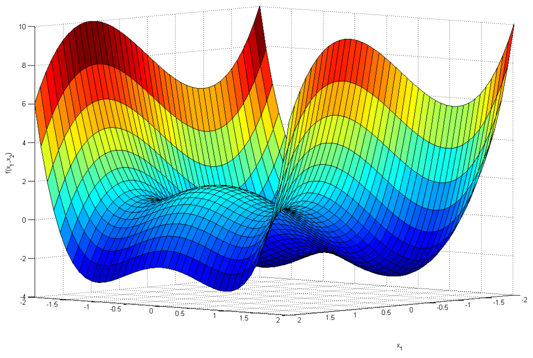


Figure 3.16: Plot of $f(x_1, x_2) = 3x_1^2 - x_1^3 - 2x_2^2 + x_2^4$, showing the various local maxima and minima of the function.

Proof: The idea behind this theorem can be stated as follows. The tangent hyperplane to the function at any extreme point must be parallel to the plane $z = 0$. This can happen if and only if the gradient $\nabla F = [\nabla^T f,\ -1]^T$ is parallel to the $z-$axis at the extreme point. Or equivalently, the gradient to the function f must be the zero vector at every extreme point, *i.e.*, $f_{x_i}(\mathbf{x}^*) = 0$ for $1 \leq i \leq n$.

To formally prove this theorem, consider the function $g_i(x_i) = f(x_1^*, x_2^*, \ldots, x_{i-1}^*, x_i, x_{i+1}^*, \ldots, x_n^*)$. If f has a local extremum at $\mathbf{x}^*$, then each function $g_i(x_i)$ must have a local extremum at x_i^*. Therefore $g_i^{'}(x_i^*) = 0$ by theorem 24. Now $g_i^{'}(x_i^*) = f_{x_i}(\mathbf{x}^*)$ so $f_{x_i}(\mathbf{x}^*) = 0$. $\square$

Applying theorem 45 to the function $f(x_1, x_2) = 9 - x_1^2 - x_2^2$, we require that at any extreme point $f_{x_1} = -2x_1 = 0 \Rightarrow x_1 = 0$ and $f_{x_2} = -2x_2 = 0 \Rightarrow x_2 = 0$. Thus, f indeed attains its maximum at the point $(0, 0)$ as shown in Figure 3.17.

Definition 37 [Critical point]: *A point $\mathbf{x}^*$ is called a critical point of a function $f(\mathbf{x})$ defined on $\mathcal{D} \subseteq \Re^n$ if*

1. *If $f_{x_i}(\mathbf{x}^*) = 0$, for $1 \leq i \leq n$.*
2. *OR $f_{x_i}(\mathbf{x}^*)$ fails to exist for any $1 \leq i \leq n$.*

A procedure for computing all critical points of a function f is:

1. Compute f_{x_i} for $1 \leq i \leq n$.

2. Determine if there are any points where any one of f_{x_i} fails to exist. Add such points (if any) to the list of critical points.

3. Solve the system of equations $f_{x_i} = 0$ simultaneously. Add the solution points to the list of saddle points.

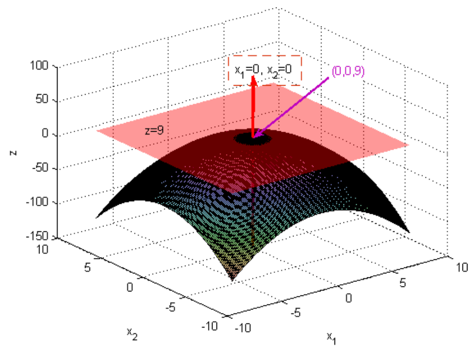


Figure 3.17: The paraboloid $f(x_1, x_2) = 9 - x_1^2 - x_2^2$ attains its maximum at $(0,0)$. The tanget plane to the surface at $(0, 0, f(0, 0))$ is also shown, and so is the gradient vector ∇F at $(0, 0, f(0, 0))$.

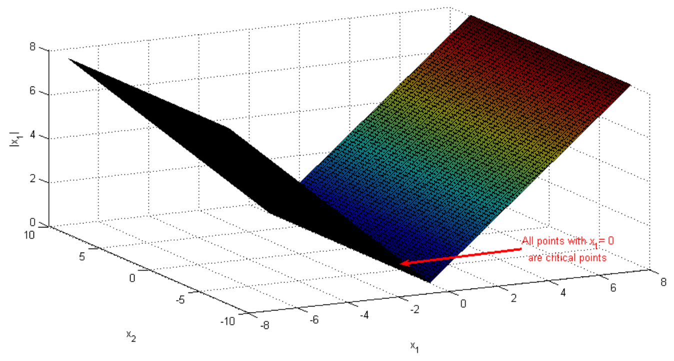


Figure 3.18: Plot illustrating critical points where derivative fails to exist.

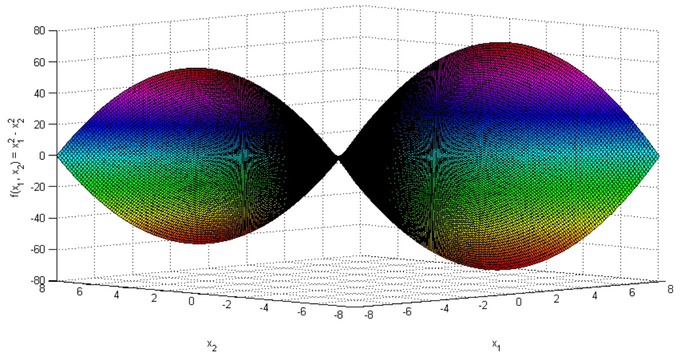


Figure 3.19: The hyperbolic paraboloid $f(x_1, x_2) = x_1^2 - x_2^2$, which has a saddle point at $(0, 0)$.

As an example, for the function $f(x_1, x_2) = |x_1|$, f_{x_1} does not exist for $(0, s)$ for any $s \in \Re$ and all of them are critical points. Figure 3.18 shows the corresponding 3−D plot.

Is the converse of theorem 45 true? That is, if you find an $\mathbf{x}^*$ that satisifes $f_{x_i}(\mathbf{x}^*) =$ for all $1 \leq i \leq n$, is it necessary that $\mathbf{x}^*$ is an extreme point? The answer is no. In fact, points that violate the converse of theorem 45 are called saddle points.

Definition 38 [Saddle point]: *A point $\mathbf{x}^*$ is called a saddle point of a function $f(\mathbf{x})$ defined on $\mathcal{D} \subseteq \Re^n$ if $\mathbf{x}^*$ is a critical point of f but $\mathbf{x}^*$ does not correspond to a local maximum or minimum of the function.*

We saw the example of a saddle point in Figure 3.7, for the case $n = 1$. The *inflection point* for a function of single variable, that was discussed earlier, is the analogue of the saddle point for a function of multiple variables. An example for $n = 2$ is the hyperbolic paraboloid[5] $f(x_1, x_2) = x_1^2 - x_2^2$, the graph of which is shown in Figure 3.19. The hyperbolic paraboloid opens up on x_1-axis (Figure 3.20 and down on x_2-axis (Figure 3.21) and has a saddle point at $(0, 0)$.

To get working on figuring out how to find the maximum and minimum of a function, we will take some examples. Let us find the critical points of $f(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 - 6x_2 + 14$ and classify the critical point. This function is a polyonomial function and is differentiable everywhere. It is a paraboloid that is shifted away from origin. To find its critical points, we will solve $f_{x_1} = 2x_1 - 2 = 0$ and $f_{x_2} = 2x_2 - 6 = 0$, which when solved simultaneously, yield a single critical point $(1, 3)$. For a simple example like this, the function f can be rewritten as $f(x_1, x_2) = (x_1 - 1)^2 + (x_2 - 3)^2 + 4$, which implies that $f(x_1, x_2) \geq 4 = f(1, 3)$. Therefore, $(1, 3)$ is indeed a local minimum (in fact a global minimum) of $f(x_1, x_2)$.

[5] The hyperbolic paraboloid is shaped like a *saddle* and can have a critical point called the saddle point.

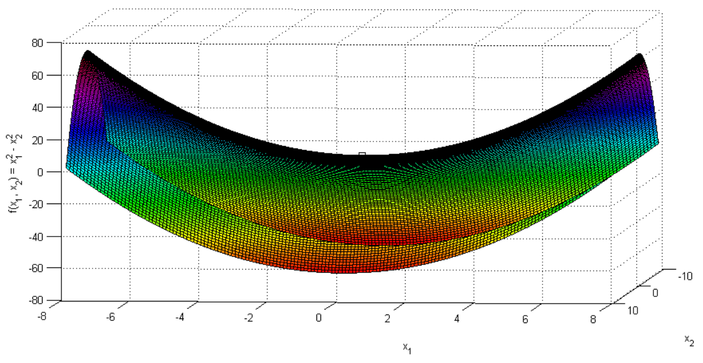


Figure 3.20: The hyperbolic paraboloid $f(x_1, x_2) = x_1^2 - x_2^2$, when viewed from the x_1 axis is concave up.

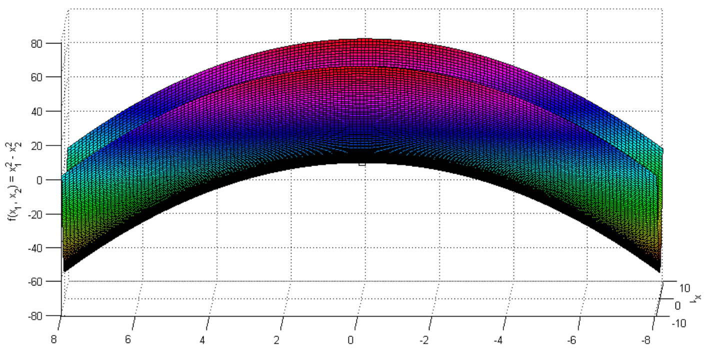


Figure 3.21: The hyperbolic paraboloid $f(x_1, x_2) = x_1^2 - x_2^2$, when viewed from the x_2 axis is concave down.

However, it is not always so easy to determine if a critical point is a point of local extreme value. To understand this, consider the function $f(x_1, x_2) = 2x_1^3 + x_1x_2^2 + 5x_1^2 + x_2^2$. The system of equations to be solved are $f_{x_1} = 6x_1^2 + x_2^2 + 10x_1 = 0$ and $f_{x_2} = 2x_1x_2 + 2x_2 = 0$. From the second equation, we get either $x_2 = 0$ or $x_1 = -1$. Using these values one at a time in the first equation, we get values for the other variables. The critical points are: $(0, 0)$, $(-\frac{5}{3}, 0)$, $(-1, 2)$ and $(-1, -2)$. Which of these critical points correspond to extreme values of the function? Since f does not have a quadratic form, it is not easy to find a lower bound on the function as in the previous example. However, we can make use of the taylor series expansion for single variable to find polynomial expansions of functions of n variables. The following theorem gives a systematic method, similar to the second derivative test for functions of single variable, for finding maxima and minima of functions of multiple variables.

Theorem 46 *Let $f : \mathcal{D} \rightarrow \Re$ where $\mathcal{D} \subseteq \Re^n$. Let $f(\mathbf{x})$ have continuous partial derivatives and continuous mixed partial derivatives in an open ball $\mathcal{R}$ containing a point $\mathbf{x}^*$ where $\nabla f(\mathbf{x}^*) = 0$. Let $\nabla^2 f(\mathbf{x})$ denote an $n \times n$ matrix of mixed partial derivatives of f evaluated at the point $\mathbf{x}$, such that the ij^{th} entry of the matrix is $f_{x_ix_j}$. The matrix $\nabla^2 f(\mathbf{x})$ is called the Hessian matrix. The Hessian matrix is symmetric*[6]. *Then,*

- *If $\nabla^2 f(\mathbf{x}^*)$ is positive definite, $\mathbf{x}^*$ is a local minimum.*
- *If $\nabla^2 f(\mathbf{x}^*)$ is negative definite (that is if $-\nabla^2 f(\mathbf{x}^*)$ is positive definite), $\mathbf{x}^*$ is a local maximum.*

Proof: Since the mixed partial derivatives of f are continuous in an open ball containing $\mathcal{R}$ containing $\mathbf{x}^*$ and since $\nabla^2 f(\mathbf{x}^*) \succ 0$, it can be shown that there exists an $\epsilon > 0$, with $\mathcal{B}(\mathbf{x}^*, \epsilon) \subseteq \mathcal{R}$ such that for all $||\mathbf{h}|| < \epsilon$, $\nabla^2 f(\mathbf{x}^* + \mathbf{h}) \succ 0$. Consider an increment vector $\mathbf{h}$ such that $(\mathbf{x}^* + \mathbf{h}) \in \mathcal{B}(\mathbf{x}^*, \epsilon)$. Define $g(t) = f(\mathbf{x}^* + t\mathbf{h}) : [0, 1] \rightarrow \Re$. Using the chain rule,

$$g'(t) = \sum_{i=1}^{n} f_{x_i}(\mathbf{x}^* + t\mathbf{h})\frac{dx_i}{dt} = \mathbf{h}^T.\nabla f(\mathbf{x}^* + t\mathbf{h})$$

Since f has continuous partial and mixed partial derivatives, g' is a differentiable function of t and

$$g''(t) = \mathbf{h}^T \nabla^2 f(\mathbf{x}^* + t\mathbf{h})\mathbf{h}$$

Since g and g' are continous on $[0, 1]$ and g' is differentiable on $(0, 1)$, we can make use of the Taylor's theorem (30) with $n = 1$ and $a = 0$ to obtain:

$$g(1) = g(0) + g'(0) + \frac{1}{2}g''(c)$$

[6]By Clairauts Theorem, if the partial and mixed derivatives of a function are continuous on an open region containing a point $\mathbf{x}^*$, then $f_{x_ix_j}(\mathbf{x}^*) = f_{x_jx_i}(\mathbf{x}^*)$, for all $i, j \in [1, n]$.

for some $c \in (0,1)$. Writing this equation in terms of f gives

$$f(\mathbf{x}^* + \mathbf{h}) = f(\mathbf{x}^*) + \mathbf{h}^T \nabla f(\mathbf{x}^*) + \frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x}^* + c\mathbf{h})\mathbf{h}$$

We are given that $\nabla f(\mathbf{x}^*) = 0$. Therefore,

$$f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*) = \frac{1}{2}\mathbf{h}^T \nabla^2 f(\mathbf{x}^* + c\mathbf{h})\mathbf{h}$$

The presence of an extremum of f at $\mathbf{x}^*$ is determined by the sign of $f(\mathbf{x}^* + \mathbf{h}) - f(\mathbf{x}^*)$. By virtue of the above equation, this is the same as the sign of $H(c) = \mathbf{h}^T \nabla^2 f(\mathbf{x}^* + c\mathbf{h})\mathbf{h}$. Because the partial derivatives of f are continuous in $\mathcal{R}$, if $H(0) \neq 0$, the sign of $H(c)$ will be the same as the sign of $H(0) = \mathbf{h}^T \nabla^2 f(\mathbf{x}^*)\mathbf{h}$ for $\mathbf{h}$ with sufficiently small components. Therefore, if $\nabla^2 f(\mathbf{x}^*)$ is positive definite, we are guaranteed to have $H(0)$ positive, implying that f has a local minimum at $\mathbf{x}^*$. Similarly, if $-\nabla^2 f(\mathbf{x}^*)$ is positive definite, we are guaranteed to have $H(0)$ negative, implying that f has a local maximum at $\mathbf{x}^*$. $\Box$

Theorem 46 gives sufficient conditions for local maxima and minima of functions of multiple variables. Along similar lines of the proof of theorem 46, we can prove necessary conditions for local extrema in theorem 47.

Theorem 47 *Let $f : \mathcal{D} \rightarrow \Re$ where $\mathcal{D} \subseteq \Re^n$. Let $f(\mathbf{x})$ have continuous partial derivatives and continuous mixed partial derivatives in an open region $\mathcal{R}$ containing a point $\mathbf{x}^*$ where $\nabla f(\mathbf{x}^*) = 0$. Then,*

- *If $\mathbf{x}^*$ is a point of local minimum, $\nabla^2 f(\mathbf{x}^*)$ must be positive semi-definite.*
- *If $\mathbf{x}^*$ is a point of local maximum, $\nabla^2 f(\mathbf{x}^*)$ must be negative semi-definite (that is, $-\nabla^2 f(\mathbf{x}^*)$ must be positive semi-definite).*

The following corollary of theorem 47 states a sufficient condition for a point to be a saddle point.

Corollary 48 *Let $f : \mathcal{D} \rightarrow \Re$ where $\mathcal{D} \subseteq \Re^n$. Let $f(\mathbf{x})$ have continuous partial derivatives and continuous mixed partial derivatives in an open region $\mathcal{R}$ containing a point $\mathbf{x}^*$ where $\nabla f(\mathbf{x}^*) = 0$. If $\nabla^2 f(\mathbf{x}^*)$ is neither positive semi-definite nor negative semi-definite (that is, some of its eigenvalues are positive and some negative), then $\mathbf{x}^*$ is a saddle point.*

Thus, for a function of more than one variable, the second derivative test generalizes to a test based on the eigenvalues of the function's Hessian matrix at the stationary point. Based on theorem 46, we will derive the second derivative test for determining extreme values of a function of two variables.

Theorem 49 *Let the partial and second partial derivatives of $f(x_1, x_2)$ be continuous on a disk with center (a, b) and suppose $f_{x_1}(a, b) = 0$ and $f_{x_2}(a, b) = 0$ so that (a, b) is a critical point of f. Let $D(a, b) = f_{x_1x_1}(a, b)f_{x_2x_2}(a, b) - [f_{x_1x_2}(a, b)]^2$. Then*[7]*,*

[7] D here stands for the discriminant.

- *If* $D > 0$ *and* $f_{x_1x_1}(a,b) > 0$*, then* $f(a,b)$ *is a local minimum.*
- *Else if* $D > 0$ *and* $f_{x_1x_1}(a,b) < 0$*, then* $f(a,b)$ *is a local maximum.*
- *Else if* $D < 0$ *then* (a,b) *is a saddle point.*

Proof: Recall the definition of positive definiteness; a matrix is positive definite if all its eigenvalues are positive. For the 2×2 matrix $\nabla^2 f$ in this problem, the product of the eigenvalues is $det(\nabla^2 f) = f_{x_1x_1}(a,b)f_{x_2x_2}(a,b) - [f_{x_1x_2}(a,b)]^2$ and the sum of the eigenvalues is $f_{x_1x_1}(a,b) + f_{x_2x_2}(a,b)$. Now:

- If $det(\nabla^2 f(a,b)) > 0$ and if additionally $f_{x_1x_1}(a,b) > 0$ (or equivalently, $f_{x_2x_2}(a,b) > 0$), the product as well as the sum of eigenvalues will be positive, implying that the eigenvalues are positive and therefore $\nabla^2 f(a,b)$ is positive definite, According to theorem 46, this is a sufficient condition for $f(a,b)$ to be a local minimum.
- If $det(\nabla^2 f(a,b)) > 0$ and if additionally $f_{x_1x_1}(a,b) < 0$ (or equivalently, $f_{x_2x_2}(a,b) < 0$), the product of the eigenvalue is positive whereas the sum is negative, implying that the eigenvalues are negative and therefore $\nabla^2 f(a,b)$ is negative definite, According to theorem 46, this is a sufficient condition for $f(a,b)$ to be a local maximum.
- If $det(\nabla^2 f(a,b)) < 0$, the eigenvalues must have opposite signs, implying that the $\nabla^2 f(a,b)$ is neither positive semi-definite nor negative-semidefinite. By corollary 48, this is a sufficient condition for $f(a,b)$ to be a saddle point.

□

We saw earlier that the critical points for $f(x_1,x_2) = 2x_1^3 + x_1x_2^2 + 5x_1^2 + x_2^2$ are $(0,0)$, $(-\frac{5}{3},0)$, $(-1,2)$ and $(-1,-2)$. To determine which of these correspond to local extrema and which are saddle, we first compute compute the partial derivatives of f:

$f_{x_1x_1}(x_1,x_2) = 12x_1 + 10$
$f_{x_2x_2}(x_1,x_2) = 2x_1 + 2$
$f_{x_1x_2}(x_1,x_2) = 2x_2$

Using theorem 49, we can verify that $(0,0)$ corresponds to a local minimum, $(-\frac{5}{3},0)$ corresponds to a local maximum while $(-1,2)$ and $(-1,-2)$ correspond to saddle points. Figure 3.22 shows the plot of the function while pointing out the four critical points.

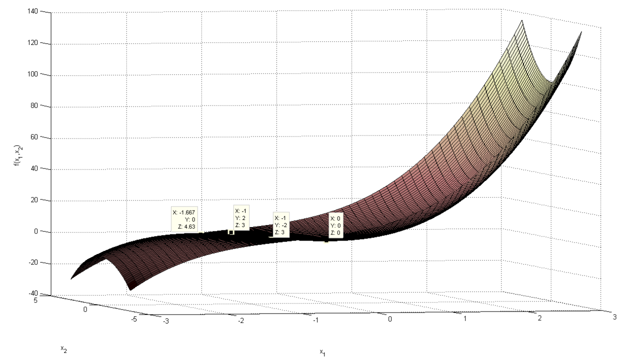


Figure 3.22: Plot of the function $2x_1^3 + x_1x_2^2 + 5x_1^2 + x_2^2$ showing the four critical points.

We will take some more examples:

1. Consider a significantly harder function $f(x, y) = 10x^2y - 5x^2 - 4y^2 - x^4 - 2y^4$. Let us find and classify its critical points. The gradient vector is $\nabla f(x, y) = [20xy - 10x - 4x^3,\ \ 10x^2 - 8y - 8y^3]$. The critical points correspond to solutions of the simultaneous set of equations

$$\begin{aligned} 20xy - 10x - 4x^3 &= 0 \\ 10x^2 - 8y - 8y^3 &= 0 \end{aligned} \tag{3.15}$$

One of the solutions corresponds to solving the system $-8y^3 + 42y - 25 = 0$[8] and $10x^2 = 50y - 25$, which have four real solutions[9], *viz.*, $(0.8567, 0.646772)$, $(-0.8567, 0.646772)$, $(2.6442, 1.898384)$, and $(-2.6442, 1.898384)$. Another real solution is $(0, 0)$. The mixed partial derivatives of the function are

$$\begin{aligned} f_{xx} &= 20y - 10 - 12x^2 \\ f_{xy} &= 20x \\ f_{yy} &= -8 - 24y^2 \end{aligned} \tag{3.16}$$

Using theorem 49, we can verify that $(2.6442, 1.898384)$ and $(-2.6442, 1.898384)$ correspond to local maxima whereas $(0.8567, 0.646772)$ and $(-0.8567, 0.646772)$ correspond to saddle points. This is illustrated in Figure 3.23.

[8]Solving this using matlab without proper scaling could give you complex values. With proper scaling of the equation, you should get $y = -2.545156$ or $y = 0.646772$ or $y = 1.898384$.

[9]The values of x corresponding to $y = -2.545156$ are complex

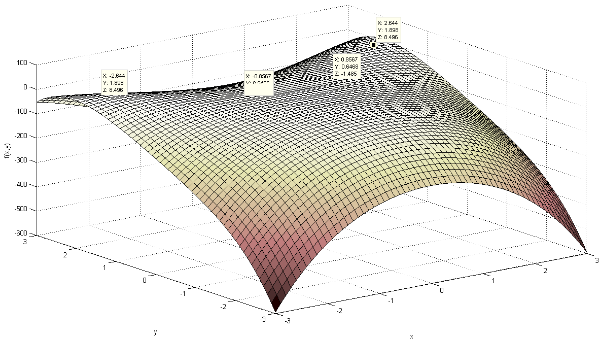


Figure 3.23: Plot of the function $10x^2y - 5x^2 - 4y^2 - x^4 - 2y^4$ showing the four critical points.

2. The function $f(x, y) = x \sin y$ has the gradient vector $[\sin y, \quad x \cos y]$. The critical points correspond to the solutions to the simultaneous set of equations

$$\begin{aligned} \sin y &= 0 \\ x \cos y &= 0 \end{aligned} \tag{3.17}$$

   The critical points are[10] $(0, n\pi)$ for $n = 0, \pm 1, \pm 2, \ldots$. The mixed partial derivatives of the function are

$$\begin{aligned} f_{xx} &= 0 \\ f_{xy} &= \cos y \\ f_{yy} &= -x \sin y \end{aligned} \tag{3.18}$$

   which tell us that the discriminant function $D = -\cos^2 y$ is always negative. Therefore, all the critical points turn out to be saddle points. This is illustrated in Figure 3.24.

Along similar lines of the single variable case, we next define the global maximum and minimum.

Definition 39 [Global maximum]: *A function f of n variables, with domain $\mathcal{D} \subseteq \Re^n$ has an absolute or global maximum at $\mathbf{x}^0$ if $\forall\ \mathbf{x} \in \mathcal{D}$, $f(\mathbf{x}) \leq f(\mathbf{x}^0)$.*

[10]Note that the *cosine* does not vanish wherever the *sin* vanishes.

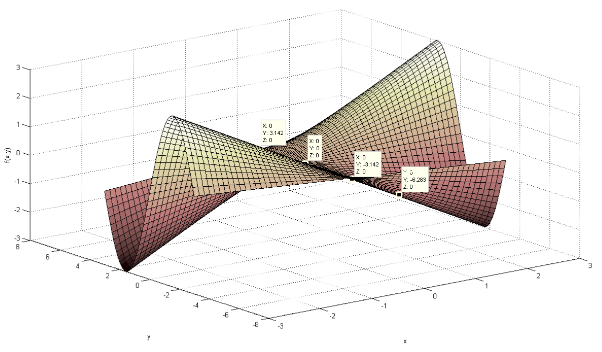


Figure 3.24: Plot of the function $x \sin y$ illustrating that all critical points are saddle points.

Definition 40 [Global minimum]: *A function f of n variables, with domain $\mathcal{D} \subseteq \Re^n$ has an absolute or global minimum at $\mathbf{x}^0$ if $\forall\ \mathbf{x} \in \mathcal{D}$, $f(\mathbf{x}) \geq f(\mathbf{x}^0)$.*

We would like to find the absolute maximum and minimum values of a function of multiple variables in a closed interval, along similar lines of the method yielded by theorem 26 for functions of single variable. The procedure was to evaluate the value of the function at the critical points as well as the end points of the interal and determine the absolute maximum and minimum values by scanning this list. To generalize the idea to function of multiple variables, we point out that the analogue of finding the value of the function at the boundaries of closed interval in the single variable case is to find the function value along the boundary curve, which reduces the evaluation of a function of multiple variables to evaluating a function of a single variable. Recall from the definitions on page 158 that a closed set in $\Re^n$ is a set that contains its boundary points (analogous to closed interval in $\Re$) while a bounded set in $\Re^n$ is a set that is contained inside a closed ball, $\mathcal{B}[\mathbf{0}, \epsilon]$. An example bounded set is $\left\{(x_1, x_2, x_3) | x_1^2 + x_2^2 + x_3^2 \leq 1\right\}$. An example unbounded set is $\{(x_1, x_2, x_3) | x_1 > 1, x_2 > 1, x_3 > 1\}$. Based on these definitions, we can state the extreme value theorem for a function of n variables.

Theorem 50 *Let $f : \mathcal{D} \rightarrow \Re$ where $\mathcal{D} \subseteq \Re^n$ is a closed bounded set and f be continuous on $\mathcal{D}$. Then f attains an absolute maximum and absolute minimum at some points in $\mathcal{D}$.*

The theorem implies that whenever a function of n variables is restricted to a bounded space, it has an absolute maximum and an absolute minimum. Following theorem 45, we note that the locally extreme values of a function occur at its critical points. By the very definition of local extremum, it cannot occur at the boundary point of $\mathcal{D}$. Since every absolute extremum is also a

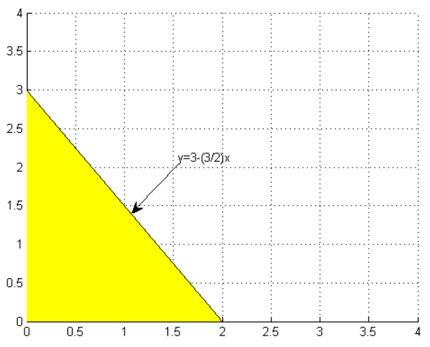


Figure 3.25: The region bounded by the points $(0, 3)$, $(2, 0)$, $(0, 0)$ on which we consider the maximum and minimum of the function $f(x, y) = 1 + 4x - 5y$.

local extremum, the absolute maximum and minimum of a function on a closed, bounded set will either happen at the critical points or at the boundary. The procedure for finding the absolute maximum and minimum of a function on a closed bounded set is outlined below and is similar to the procedure 4 for a function of single variable continuous on a closed and bounded interval:

Procedure 5 [Finding extreme values on closed, bounded sets]: *To find the absolute maximum and absolute minimum of a continuous function f on a closed bounded set $\mathcal{D}$;*

- *evaluate f at the critical points of f on $\mathcal{D}$*
- *find the extreme values of f on the boundary of $\mathcal{D}$*
- *the largest of the values found above is the absolute maximum, and the smallest of them is the absolute mininum.*

We will take some examples to illustrate procedure 5.

1. Consider the function $f(x, y) = 1 + 4x - 5y$ defined on the region $\mathcal{R}$ bounded by the points $(0, 3)$, $(2, 0)$, $(0, 0)$. The region $\mathcal{R}$ is shown in Figure 3.25 and is bounded by three line segments

   - $\mathbf{B}_1$: $x = 0$, $0 \leq y \leq 3$
   - $\mathbf{B}_2$: $y = 0$, $0 \leq x \leq 2$
   - and $\mathbf{B}_3$: $y = 3 - \frac{3}{2}x$, $0 \leq x \leq 2$.

   The linear function $f(x, y) = 1 + 4x - 5y$ has no critical points, since $\nabla f(x, y) = [4, \;\; 5]^T$ is defined everywhere, though it cannot disappear at any point. In fact, linear functions have no critical points and the extreme values are always assumed at the boundaries; this forms the basis of linear programming. We will find the extreme values on the boundaries.

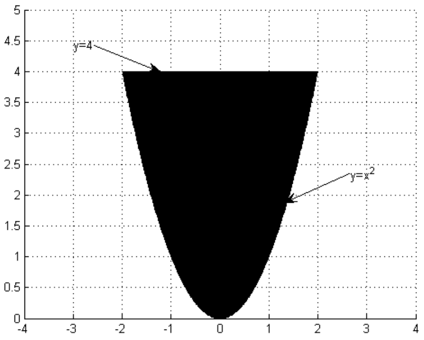


Figure 3.26: The region $\mathcal{R}$ bounded by $y = x^2$ and $y = 4$ on which we consider the maximum and minimum of the function $f(x, y) = 1 - xy - x - y$.

- On $\mathbf{B}_1$, $f(x, y) = f(0, y) = 1 - 5y$, for $y \in [0, 3]$. This is a single variable extreme value problem for a continuous function. Its largest value is assumed at $y = 0$ and equals 1 while the smallest value is assumed at $y = 3$ and equals -14.
- On $\mathbf{B}_2$, $f(x, y) = f(x, 0) = 1 + 4x$, for $x \in [0, 2]$. This is again a single variable extreme value problem for a continuous function. Its largest value is assumed at $x = 2$ and equals 9 while the smallest value is assumed at $x = 0$ and equals 1.
- On $\mathbf{B}_3$, $f(x, y) = 1 + 4x - 5\,(3 - (3/2)x) = -14 + (23/2)x$, for $x \in [0, 2]$. This is also a single variable extreme value problem for a continuous function. Its largest value is assumed at $x = 2$ and equals 9 while the smallest value is assumed at $x = 0$ and equals -14.

Thus, the absolute maximum is attained by f at $(2, 0)$ while the absolute minimum is attained at $(0, 3)$. Both extrema are at the vertices of the polygon (triangle) This example illustrates the general procedure for determining the absolute maximum and minimum of a function on a closed, bounded set. However, the problem can become very hard in practice as the function f gets complex.

2. Let us look at a harder problem. Let us find the absolute maximum and the absolute minimum of the function $f(x, y) = 1 - xy - x - y$ on the region $\mathcal{R}$ bounded by $y = x^2$ and $y = 4$. This is not a linear function any longer. The region $\mathcal{R}$ is shown in Figure 3.26 and is bounded by

   - $\mathbf{B}_1$: $y = x^2$, $-2 \le x \le 2$
   - $\mathbf{B}_2$: $y = 4$, $-2 \le x \le 2$

   Since $f(x, y) = 1 - xy - x - y$ is differentiable everywhere, the critical point of f is characterized by $\nabla f(x, y) = [-y - 1,\ x - 1]^T = \mathbf{0}$, that is

$x = -1, \quad y = -1$. However, this point does not lie in $\mathcal{R}$ and hence, there are no critical points, in $\mathcal{R}$. Along similar lines of the previous problem, we will find the extreme values of f on the boundaries of $\mathcal{R}$.

- On $\mathbf{B}_1$, $f(x, y) = 1 - x^3 - x - x^2$, for $x \in [-2, 2]$. This is a single variable extreme value problem for a continuous function. Its critical points correspond to solutions of $3x^2 + 2x + 1 = 0$. However, this equation has no real solutions[11] and therefore, the function's extreme values are only at the boundary points; the minimum value -13 is attained at $x = 2$ and the maximum value 7 is attained at $x = -2$.
- On $\mathbf{B}_2$, $f(x, y) = 1 - 4x - x - 4 = -3 - 5x$, for $x \in [-2, 2]$. This is again a single variable extreme value problem for a continuous function. It has no critical points and extreme values correspond to the boundary points; its maximum value 7 is assumed at $x = -12$ while the minimum value -13 is assumed at $x = 2$.

Thus, the absolute maximum value 7 is attained by f at $(-2, 4)$ while the absolute minimum value -13 is attained at $(2, 4)$.

3. Consider the same problem as the previous one, with a slightly different objective function, $f(x, y) = 1 + xy - x - y$. The critical point of f is characterized by $\nabla f(x, y) = [y - 1, \quad x - 1]^T = \mathbf{0}$, that is $x = 1, \quad y = 1$. This lies within $\mathcal{R}$ and f takes the value 0 at $(1, 1)$. Next, we find the extreme values of f on the boundaries of $\mathcal{R}$.

   - On $\mathbf{B}_1$, $f(x, y) = 1 + x^3 - x - x^2$, for $x \in [-2, 2]$. Its critical points correspond to solutions of $3x^2 - 2x - 1 = 0$. Its solutions are $x = 1$ and $x = -\frac{1}{3}$. The function values corresponding to these points are $f(1, 1) = 0$ and $f(-1/3, 1/9) = 32/27$. At the boundary points, the function assumes the values $f(-2, 4) = -9$ and $f(2, 4) = 3$. Thus, the maximum value on $\mathbf{B}_1$ is $f(2, 4) = 3$ and the minimum value is $f(-2, 4) = -9$.
   - On $\mathbf{B}_2$, $f(x, y) = 1 + 4x - x - 4 = -3 + 3x$, for $x \in [-2, 2]$. It has no critical points and extreme values correspond to the boundary points; At the boundary points, the function assumes the values $f(-2, 4) = -9$ and $f(2, 4) = 3$, which correspond to the minimum and maximum values respectively of f on $\mathbf{B}_2$.

   Thus, the absolute maximum value 3 is attained by f at $(2, 4)$ while the absolute minimum value -9 is attained at $(-2, 4)$.

3.1.5 Absolute extrema and Convexity

Theorem 46 specified a sufficient condition for the local minimum of a differentiable function with continuous partial and mixed partial derivatives, while

[11]The complex solutions are $x = -\frac{1}{3} + i\frac{1}{3}\sqrt{2}$ and $x = -\frac{1}{3} - i\frac{1}{3}\sqrt{2}$.

theorem 47 specified a necessary condition for the same. Can these conditions be extended to globally optimal solutions? The answer is that the extensions to globally optimal solutions can be made for a specific class of optimization problems called convex optimization problems. In the next section we introduce the concept of convex sets and convex functions, enroute to discussing convex optimization.

3.2 Convex Optimization Problem

A function $f(.)$ is called convex if its value at the scalar combination of two points x and y is less than the same scalar combination of the function at the two points. In other words, $f(.)$ is convex if and only if:

$$\begin{aligned}&f(\alpha x + \beta y) \le \alpha f(x) + \beta f(y)\\&\text{if } \alpha + \beta = 1, \alpha \ge 0, \beta \ge 0\end{aligned} \tag{3.19}$$

For a convex optimization problem, the objective function $f(x)$ as well as the inquality functions $g_i(x), i = 1, \dots, m$ are convex. The equality constraints are linear, *i.e.*, of the form, $Ax = b$.

$$\begin{array}{ll}\text{minimize} & f(x)\\ \text{subject to} & g_i(x) \le 0, \quad i = 1, \dots, m\\ & Ax = b\end{array} \tag{3.20}$$

Least squares and linear programming are special cases of convex optimization problems. Like in the case of linear programming, there are no analytical solutions for convex optimization problems. But they can be solved reliably, efficiently and optimally. There are not many well developed software for the general class of convex optimization problems, though there are several software packages in matlab, C, *etc.*, and many free softwares as well. The computation time is polynomial but more complicated to be expressed exactly because the computation time depends on the cost of validating the function values and their derivates. Modulo that, computation time for convex optimization problems is similar to that for linear programming problems.

To pose pratical problems as convex optimization problems is more difficult than to recognize least squares and linear programs. There exist many techniques to reformulate problems in the convex form. However, surprisingly, many problems in practice can be solved via convex optimization.

3.2.1 Why Convex Optimization?

We will see in this sequel, that generic convex programs, under mild computability and boundedness assumptions, are computationally tractable. Many convex programs admit theoretically and practically efficient solution methods. Convex optimization admits *duality theory*, which can be used to quantitatively establish the quality of an approximate solution. Even though duality may not yield a closed-form solution, it often facilitates nontrivial reformulations of the problem. Duality theory also comes handy in confirming if an approximate solution is optimal.

In contrast to this, rarely does it happen that a global solution can be efficiently found for nonconvex optimization programs[12]. For most nonconvex programs, there are no sound techniques for certifying the global optimalality of approximate solutions or estimating how non-optimal an approximate solution is.

3.2.2 History

Numerical optimization started in the 1940s with the development of the simplex method for linear programming. The next obvious extension to linear programming was by replacing the linear cost function with a quadratic cost function. The linear inequality constraints were however maintained. This first extension took place in the 1950s. We can expect that the next step would have been to replace the linear constraints with quadratic constraints. But it did not really happen that way. On the other hand, around the end of the 1060s, there was another non-linear, convex extension of linear programming called *geometric programming*. Geometric programming includes linear programming as a special case. Nothing more happened until the beginning of the 1990s. The beginning of the 1990s was marked by a big explosion of activities in the area of convex optimizations, and development really picked up. Researches formulated different and more general classes of convex optimization problems that are known as semidefinite programming, second-order cone programming, quadratically constrained quadratic programming, sum-of-squares programming, *etc.*

The same happened in terms of applications. Since 1990s, applications have been investigated in many different areas. One of the first application areas was control, and the optimization methods that were investigated included semidefinite programming for certain control problem. Geometric programming had been around since late 1960s and it was applied extensively to circuit design problems. Quadratic programming found application in machine learning problem formulations such as support vector machines. Semi-definite programming relaxations found use in combinatorial optimization. There were many other interesting applications in different areas such as image processing, quantum information, finance, signal processing, communications, *etc.*

This first look at the activities involving applications of optimization clearly indicates that a lot a of development took place around the 1990s. Further, people extended interior-point methods (which were already known for linear

[12]Optimization problems such as singular value decomposition are some few exceptions to this.

programming since 1984[13]) to non-linear convex optimization problems. A highlight in this area was the work of Nesterov and Nemirovski who extended Karmarkar's work to polynomial-time interior-point methods for nonlinear convex programming in their book published in 1994, though the work actually took place in 1990. As a result, people started looking at non-linear convex optimization in a special way; instead of treating non-linear convex optimization as a special case of non-linear optimization, they looked at it as an extension of linear programming which can be solved almost with the same efficiency. Once people started looking at applications of non-linear convex optimization, they discovered many!

We will begin with a background on convex sets and functions. Convex sets and functions constitute the basic theory for the entire area of convex optimization. Next, we will discuss some standard problem classes and some recently studied problem classes such as semi-definite programming and cone programming. FInally, we will look at applications.

3.2.3 Affine Set

Definition 41 [Affine Set]: *A set $\mathcal{A}$ is called affine if the line connecting any two distinct points in the set is completely contained within $\mathcal{A}$. Mathematically, the set $\mathcal{A}$ is called affine if*

$$\forall\ \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{A}, \quad \theta \in \Re \quad \Rightarrow \quad \theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \mathcal{A} \tag{3.21}$$

Theorem 51 *The solution set of the system of linear equations $A\mathbf{x} = \mathbf{b}$ is an affine set.*

Proof: Suppose $\mathbf{x}_1$ and $\mathbf{x}_2$ are solutions to the system $A\mathbf{x} = \mathbf{b}$ with $\mathbf{x}_1 \neq \mathbf{x}_2$. Then, $A\left(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2\right) = \theta\mathbf{b} + (1-\theta)\mathbf{b} = \mathbf{b}$. Thus, $\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \mathcal{A}$, implying that the solution set of the system $A\mathbf{x} = \mathbf{b}$ is an affine set. $\square$

In fact, converse of theorem 51 is also true; any affine set can be expressed as the solution set of a system of linear equations $A\mathbf{x} = \mathbf{b}$.

3.2.4 Convex Set

Definition 42 [Convex Set]: *A set $\mathcal{C}$ is called convex if the line segment connecting any two points in the set is completely contained within $\mathcal{C}$. Else $\mathcal{C}$ is called concave. That is,*

$$\forall\ \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C} \quad 0 \leq \theta \leq 1 \quad \Rightarrow \quad \theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \mathcal{C} \tag{3.22}$$

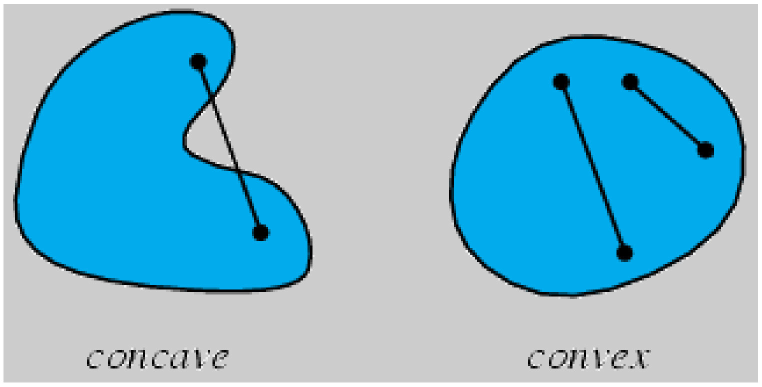


Figure 3.27: Examples of convex and non-convex sets.

Figure 3.27 shows examples of convex and non-convex (concave) sets. Since an affine set contains any line passing through two distinct points in the set, it also contains any line segment connecting two points in the set. Thus, an affine set is our first example of a convex set.

A set $\mathcal{C}$ is a convex cone if it is covex and additionally, for every point $\mathbf{x} \in \mathcal{C}$, all non-negative multiples of $\mathbf{x}$ are also in $\mathcal{C}$. In other words,

$$\forall \mathbf{x}_1, \mathbf{x}_2 \in C \quad \theta_1, \theta_2 \geq 0 \quad \Rightarrow \quad \theta_1\mathbf{x}_1 + \theta_2\mathbf{x}_2 \in \mathcal{C} \tag{3.23}$$

Combinations of the form $\theta_1\mathbf{x}_1 + \theta_2\mathbf{x}_2$ for $\theta_1 \geq 0, \theta_2 \geq 0$ are called conic combinations. We will state a related definition next - that of the convex hull of a set of points.

Definition 43 [Convex Hull]: *A convex combination of the set of points $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ is any point $\mathbf{x}$ of the form*

$$\mathbf{x} = \sum_{i=1}^{k} \theta_i \mathbf{x}_i \quad \text{with} \sum_{i=1}^{k} \theta_i = 1 \quad \text{and } \theta_i \geq 0 \tag{3.24}$$

The convex hull conv($\mathcal{S}$) of the set of points $\mathcal{S}$ is the set of all convex combinations of points in $\mathcal{S}$. The convex hull of a convex set $\mathcal{S}$ is $\mathcal{S}$ itself.

[13]The first practical polynomial time algorithm for linear programming by Karmarkar (1984) involved interior-point methods.

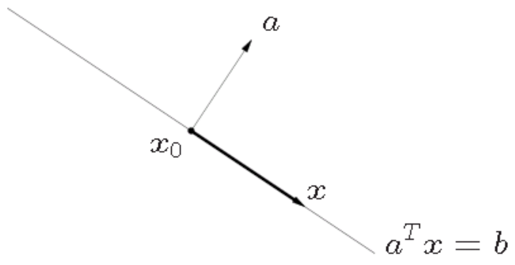


Figure 3.28: Example of a hyperplane in $\Re^2$.

3.2.5 Examples of Convex Sets

We will look at simple but important examples of convex sets. We will also look at some operations that preserve convexity.

A *hyperplane* is the most common example of a convext set. A hyperplane is the set of solutions to a linear system of equations of the form $a^T\mathbf{x} = b$ with $a \neq 0$ and was defined earlier in definition 34. A *half space* is a solution set over the linear inequality $a^T\mathbf{x} \leq b, a \neq 0$. The hyperplane $a^T\mathbf{x} = b$ bounds the half-space from one side.

Formally,

Hyperplane: $\{\mathbf{x}|a^T\mathbf{x} = b, a \neq 0\}$. Figure 3.28 shows an example hyperplane in $\Re^2$. a is the normal vector.

Halfspace: $\{\mathbf{x}|a^T\mathbf{x} \leq b, a \neq 0\}$. Figure 3.29 shows an example half-space in $\Re^2$.

The hyperplane is convex and affine, whereas the halfspace is merely convex and not affine.

Another simple example of a convex set is a *closed ball* in $\Re^n$ with radius r and center $\mathbf{x}_c$ which is an n-dimensional vector.

$$\mathcal{B}[\mathbf{x}_c, r] = \{\mathbf{x}_c + r\mathbf{u} \mid ||\mathbf{u}||_2 \leq 1\}$$

where $\mathbf{u}$ is a vector with norm less than or equal to 1. The open ball $\mathcal{B}(\mathbf{x}_c, r)$ is also convex. Replacing r with a non-singular square matrix A, we get an *ellipsoid* given by

$$\{\mathbf{x}_c + A\mathbf{u} \mid ||\mathbf{u}||_2 \leq 1\}$$

which is also a convex set. Another equivalent representation of the ellipsoid can be obtained by observing that for any point $\mathbf{x}$ in the ellipsoid, $||A^{-1}(\mathbf{x}-\mathbf{x}_c)||_2 \leq 1$, that is $(\mathbf{x} - \mathbf{x}_c)^T(A^{-1})^TA^{-1}(\mathbf{x} - \mathbf{x}_c) \leq 1$. Since $(A^{-1})^T = (A^T)^{-1}$ and

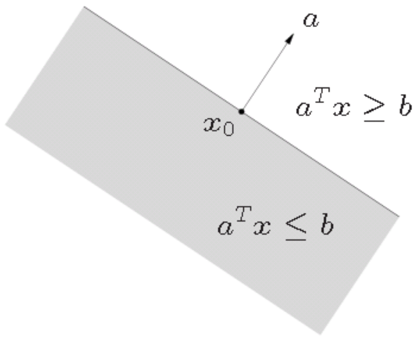


Figure 3.29: Example of a half-space in $\Re^2$.

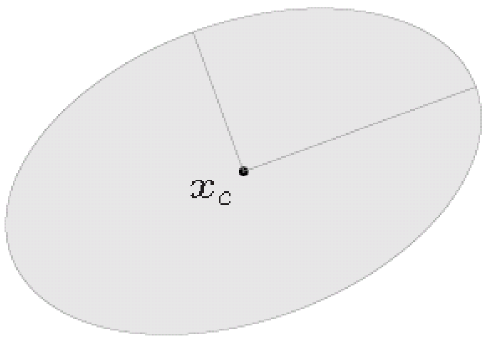


Figure 3.30: Example of a ellipsoid in $\Re^2$.

$A^{-1}B^{-1} = (BA)^{-1}$, the ellipsoid can be equivalently defined as $\left\{\mathbf{x} | (\mathbf{x} - \mathbf{x}_c)^T P^{-1}(\mathbf{x} - \mathbf{x}_c) \leq 1\right\}$ where $P = (AA^T)$ is a symmetric matrix. Furthermore, P is positive definite, since A is non-singular (*c.f.* page 151).

Matrix A determines the size of the ellipsoid; the eigenvalue λ_i of A determines the length of the i^{th} semi-axis of the ellipsoid (see page number 149). The ellipsoid is another example of a convex set and is a generalization of the eucledian ball. Figure 3.30 illustrates an ellipsoid in $\Re^2$.

A *norm ball* is a ball with an arbitrary norm. A norm ball with center $\mathbf{x}_c$ and radius r is given by

$$\{\mathbf{x} \mid ||\mathbf{x} - \mathbf{x}_c|| \leq r\}$$

By the definition of the norm, a ball in that norm will be convex. The norm ball with the $\infty-$norm corresponds to a square in $\Re^2$, while the norm ball with the $1-$norm in $\Re^2$ corresponds to the same square rotated by $45°$. The norm ball is convex for all norms.

The definition of cone can be extended to any arbitrary norm to define a

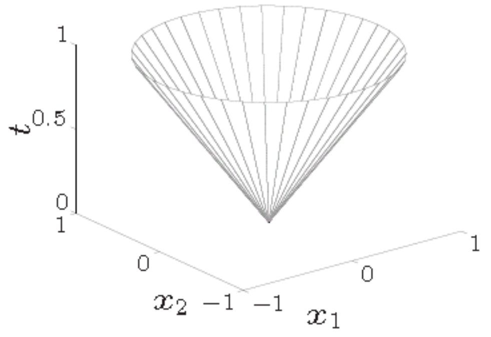


Figure 3.31: Example of a cone in $\Re^2$.

norm cone. The set of all pairs $(\mathbf{x}, t)$ satisfying $||\mathbf{x}|| \leq t$, *i.e.*,

$$\{(\mathbf{x}, t) \mid ||\mathbf{x}|| \leq t\}$$

is called a norm cone

When the norm is the eucledian norm, the cone (which looks like an ice-cream cone) is called the *second order cone.* Norm cones are always convex. Figure 3.31 shows a cone in $\Re^2$. In general, the cross section of a norm cone has the shape of a norm ball with the same norm. The norm cone for the $\infty-$norm is a square pyramid in $\Re^3$ and the cone for $1-$norm in $\Re^3$ is the same square pyramid rotated by $45°$.

A *polyhedron* is another convex set which is given as the solution set of a finite set of linear equalities and inequalities. In matrix form, the inequalities can be stated as

$$\begin{aligned} A\mathbf{x} \preceq \mathbf{b} \quad & A \in \Re^{m \times n} \\ C\mathbf{x} = \mathbf{d} \quad & C \in \Re^{p \times n} \end{aligned} \tag{3.25}$$

where $\preceq$ stands for component-wise inequality of the form $\leq$[14]. A polyhedron can also be represented as the intersection of a finite number of halfspaces and hyperplanes. Figure 3.32 depicts a typical polyhedron in $\Re^2$. An affine set is a special type of polyhedron.

A last simple example is the positive semi-definite cone. Let $\mathcal{S}^n$ be the set of all symmetric $n \times n$ matrices and $\mathcal{S}^n_+ \subset \mathcal{S}^n$ be the set of all positive semi-definite $n \times n$ matrices. The set $\mathcal{S}^n_+$ is a convex cone and is called the *positive semi-definite cone.* Consider a positive semi-definite matrix S in $\Re^2$. Then S must of the form

[14]The component-wise inequality corresponds to a generalized inequality $\preceq_K$ with $K = \Re^n_+$.

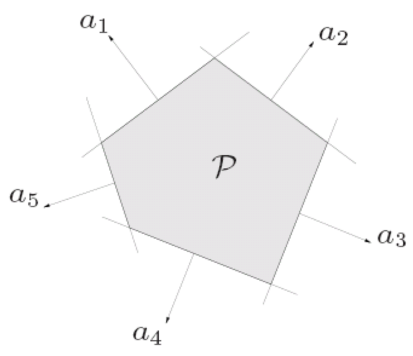


Figure 3.32: Example of a polyhedron in $\Re^2$.

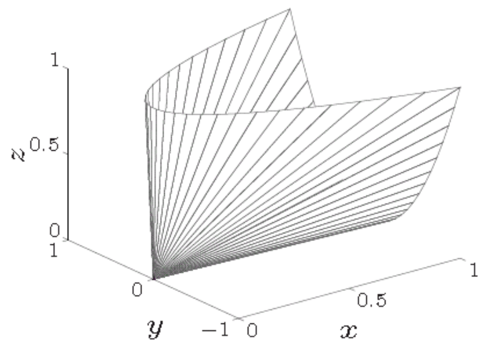


Figure 3.33: Example of a positive semidefinite cone in $\Re^2$.

$$S = \begin{bmatrix} x & y \\ y & z \end{bmatrix} \tag{3.26}$$

We can represent the space of matrices $\mathcal{S}_+^2$ of the form $S \in \mathcal{S}_+^2$ as a three dimensional space with non-negative x, y and z coordinates and a non-negative determinant. This space corresponds to a cone as shown in Figure 3.33.

3.2.6 Convexity preserving operations

In practice if you want to establish the convexity of a set $\mathcal{C}$, you could either

1. prove it from first principles, *i.e.*, using the definition of convexity or
2. prove that $\mathcal{C}$ can be built from simpler convex sets through some basic operations which preserve convexity.

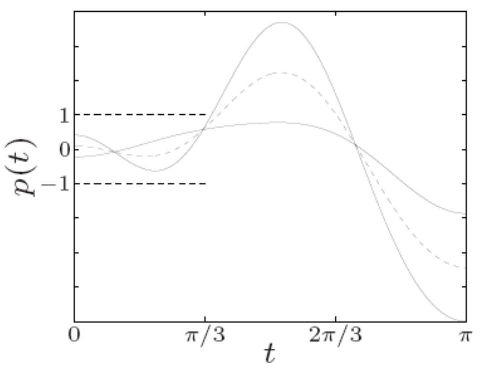


Figure 3.34: Plot for the function in (3.28)

Some of the important operations that preserve complexity are:

Intersection

The intersection of any number of convex sets is convex[15]. Consider the set $\mathcal{S}$:

$$\mathcal{S} = \left\{ \mathbf{x} \in \Re^n \mid |p(t)| \leq 1 \ for \ |t| \leq \frac{\pi}{3} \right\} \tag{3.27}$$

where

$$p(t) = x_1 \cos t + x_2 \cos 2t + \ldots + x_m \cos mt \tag{3.28}$$

Any value of t that satisfies $|p(t)| \leq 1$, defines two regions, *viz.*,

$$\Re^{\leq}(t) = \{\mathbf{x} \mid x_1 \cos t + x_2 \cos 2t + \ldots + x_m \cos mt \leq 1\}$$

and

$$\Re^{\geq}(t) = \{\mathbf{x} \mid x_1 \cos t + x_2 \cos 2t + \ldots + x_m \cos mt \geq -1\}$$

Each of the these regions is convex and for a given value of t, the set of points that may lie in $\mathcal{S}$ is given by

$$\Re(t) = \Re^{\leq}(t) \cap \Re^{\geq}(t)$$

This set is also convex. However, not all the points in $\Re(t)$ lie in $\mathcal{S}$, since the points that lie in $\mathcal{S}$ satisfy the inequalities for every value of t. Thus, $\mathcal{S}$ can be given as:

$$\mathcal{S} = \cap_{|t| \leq \frac{\pi}{3}} \Re(t)$$

[15]Exercise: Prove.

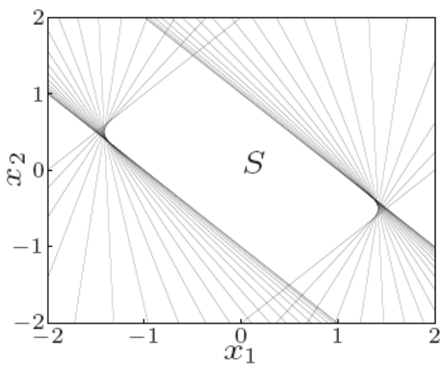


Figure 3.35: Illustration of the closure property for $\mathcal{S}$ defined in (3.27), for $m = 2$.

Affine transform

An affine transform is one that preserves

- Collinearity between points, *i.e.*, three points which lie on a line continue to be collinear after the transformation.
- Ratios of distances along a line, *i.e.*, for distinct colinear points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$, $\frac{||\mathbf{p}_2 - \mathbf{p}_1||}{||\mathbf{p}_3 - \mathbf{p}_2||}$ is preserved.

An affine transformation or affine map between two vector spaces $f : \Re^n \rightarrow \Re^m$ consists of a linear transformation followed by a translation:

$$\mathbf{x} \mapsto A\mathbf{x} + \mathbf{b}$$

where $A \in \Re^{n \times m}$ and $\mathbf{b} \in \Re^m$. In the finite-dimensional case each affine transformation is given by a matrix A and a vector $\mathbf{b}$.

The image and pre-image of convex sets under an affine transformation defined as

$$f(\mathbf{x}) = \sum_{i}^{n} x_i a_i + b$$

yield convex sets[16]. Here a_i is the i^{th} row of A. The following are examples of convex sets that are either images or inverse images of convex sets under affine transformations:

1. the solution set of linear matrix inequality $(A_i, B \in \mathcal{S}^m)$

$$\{\mathbf{x} \in \Re^n \mid x_1 A_1 + \ldots + x_n A_n \preceq B\}$$

[16]Exercise: Prove.

is a convex set. Here $A \preceq B$ means $B - A$ is positive semi-definite[17]. This set is the inverse image under an affine mapping of the positive semi-definite cone. That is, $f^{-1}(cone) = \left\{\mathbf{x} \in \Re^n \mid B - (x_1A_1 + \ldots + x_nA_n) \in \mathcal{S}_+^m\right\} = \left\{\mathbf{x} \in \Re^n | B \geq (x_1A_1 + \ldots + x_nA_n)\right\}$.

2. hyperbolic cone ($P \in \mathcal{S}_+^n$), which is the inverse image of the norm cone $\mathcal{C}_{m+1} = \{(\mathbf{z}, u) | ||\mathbf{z}|| \leq u, u \geq 0, \mathbf{z} \in \Re^m\} = \left\{(\mathbf{z}, u) | \mathbf{z}^T\mathbf{z} - u^2 \leq 0,\ u \geq 0, \mathbf{z} \in \Re^m\right\}$ is a convex set. The inverse image is given by $f^{-1}(\mathcal{C}_{m+1}) = \left\{\mathbf{x} \in \Re^n \mid \left(A\mathbf{x}, \mathbf{c}^T\mathbf{x}\right) \in \mathcal{C}_{m+1}\right\} = \left\{\mathbf{x} \in \Re^n | \mathbf{x}^TA^TA\mathbf{x} - (\mathbf{c}^T\mathbf{x})^2 \leq 0\right\}$. Setting, $P = A^TA$, we get the equation of the hyperbolic cone:

$$\left\{\mathbf{x} \mid \mathbf{x}^TP\mathbf{x} \leq (\mathbf{c}^T\mathbf{x})^2, \mathbf{c}^T\mathbf{x} \geq 0\right\}$$

Perspective and linear-fractional functions

The perspective function $P : \Re^{n+1} \rightarrow \Re^n$ is defined as follows:

$$\begin{aligned} &P : \Re^{n+1} \rightarrow \Re^n \text{ such that} \\ &P(x, t) = x/t \qquad \qquad \textbf{dom } P = \{(x, t) \mid t > 0\} \end{aligned} \tag{3.29}$$

The linear-fractional function f is a generalization of the perspective function and is defined as: $\Re^n \rightarrow \Re^m$:

$$\begin{aligned} &f : \Re^n \rightarrow \Re^m \text{ such that} \\ &f(\mathbf{x}) = \frac{A\mathbf{x}+\mathbf{b}}{\mathbf{c}^T\mathbf{x}+d} \qquad \qquad \textbf{dom } f = \{\mathbf{x} \mid \mathbf{c}^T\mathbf{x} + d > 0\} \end{aligned} \tag{3.30}$$

The images and inverse images of convex sets under perspective and linear-fractional functions are convex[18].

Consider the linear-fractional function $f = \frac{1}{x_1+x_2+1}x$. Figure **??** shows an example convex set. Figure **??** shows the image of this convex set under the linear-fractional function f.

Supporting Hyperplane Theorem

On page 3.1.4, we introduced the concept of the hyperplane. For disjoint convex sets, we state the *separating hyperplane theorem.*

Theorem 52 *If $\mathcal{C}$ and $\mathcal{D}$ are disjoint convex sets, i.e., $\mathcal{C} \cap \mathcal{D} = \phi$, then there exists $\mathbf{a} \neq \mathbf{0}$, with a $b \in \Re$ such that*
$\mathbf{a}^T\mathbf{x} \leq \mathbf{b}$ *for* $\mathbf{x} \in \mathcal{C}$,
$\mathbf{a}^T\mathbf{x} \geq \mathbf{b}$ *for* $\mathbf{x} \in \mathcal{D}$.

That is, the hyperplane $\left\{\mathbf{x} | \mathbf{a}^T\mathbf{x} = \mathbf{b}\right\}$ separates $\mathcal{C}$ and $\mathcal{D}$. The seperating hyperplane need not be unique though.

[17]The inequality induced by positive semi-definiteness corresponds to a generalized inequality $\preceq_K$ with $K = \mathcal{S}_+^n$.

[18]Exercise: Prove.

Proof: We first note that the set $\mathcal{S} = \{\mathbf{x} - \mathbf{y} | \mathbf{x} \in \mathcal{C}, \mathbf{y} \in \mathcal{D}\}$ is convex, since it is the sum of two convex sets. Since $\mathcal{C}$ and $\mathcal{D}$ are disjoint, $\mathbf{0} \notin \mathcal{S}$. Consider two cases:

1. Suppose $\mathbf{0} \notin closure(\mathcal{S})$. Let $\mathcal{E} = \{0\}$ and $\mathcal{F} = closure(\mathbf{S})$. Then, the euclidean distance between $\mathcal{E}$ and $\mathcal{F}$, defined as

   $dist(\mathcal{E};\mathcal{F}) = inf\left\{||\mathbf{u} - \mathbf{v}||_2 | \mathbf{u} \in \mathcal{E}, \mathbf{v} \in \mathcal{F}\right\}$

   is positive, and there exists a point $\mathbf{f} \in \mathcal{F}$ that achieves the minimum distance, i.e., $||\mathbf{f}||_2 = dist(\mathcal{E},\mathcal{F})$. Define $\mathbf{a} = \mathbf{f}$, $\mathbf{b} = ||\mathbf{f}||_2$. Then $\mathbf{a} \neq \mathbf{0}$ and the affine function $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x} - b = \mathbf{f}^T(\mathbf{x} - \frac{1}{2}\mathbf{f})$ is nonpositive on $\mathcal{E}$ and nonnegative on $\mathcal{F}$, *i.e.*, that the hyperplane $\left\{\mathbf{x}|\mathbf{a}^T\mathbf{x} = b\right\}$ separates $\mathcal{E}$ and $\mathcal{F}$. Thus, $\mathbf{a}^T(\mathbf{x} - \mathbf{y}) > 0$ for all $\mathbf{x} - \mathbf{y} \in \mathcal{S} \subseteq closure(\mathcal{S})$, which implies that, $\mathbf{a}^T\mathbf{x} \geq \mathbf{a}^T\mathbf{y}$ for all $\mathbf{x} \in \mathcal{C}$ and $\mathbf{y} \in \mathcal{D}$.

2. Suppose, $0 \in closure(\mathcal{S})$. Since $0 \notin \mathcal{S}$, it must be in the boundary of $\mathcal{S}$.

   - If $\mathcal{S}$ has empty interior, it must lie in an affine set of dimension less than n, and any hyperplane containing that affine set contains $\mathcal{S}$ and is a hyperplane. In other words, $\mathcal{S}$ is contained in a hyperplane $\left\{\mathbf{z}|\mathbf{a}^T\mathbf{z} = b\right\}$, which must include the origin and therefore $b = 0$. In other words, $\mathbf{a}^T\mathbf{x} = \mathbf{a}^T\mathbf{y}$ for all $\mathbf{x} \in \mathcal{C}$ and all $\mathbf{y} \in \mathcal{D}$ gives us a trivial separating hyperplane.
   - If $\mathcal{S}$ has a nonempty interior, consider the set

     $\mathcal{S}_{-\epsilon} = \{\mathbf{z}|B(\mathbf{z},\epsilon) \subseteq \mathcal{S}\}$

     where $B(\mathbf{z},\epsilon)$ is the Euclidean ball with center $\mathbf{z}$ and radius $\epsilon > 0$. $\mathcal{S}_{-\epsilon}$ is the set $\mathcal{S}$, shrunk by ϵ. $closure\left(\mathcal{S}_{-\epsilon}\right)$ is closed and convex, and does not contain $\mathbf{0}$, so as argued before, it is separated from $\{\mathbf{0}\}$ by atleast one hyperplane with normal vector $\mathbf{a}(\epsilon)$ such that

     $\mathbf{a}(\epsilon)^T\mathbf{z} \geq 0$ for all $\mathbf{z} \in \mathcal{S}_\epsilon$

     Without loss of generality assume $||\mathbf{a}(\epsilon)||_2 = 1$. Let ϵ_k, for $k = 1, 2, \ldots$ be a sequence of positive values of ϵ_k with $\lim\limits_{k\to\infty} \epsilon_k = 0$. Since $||\mathbf{a}(\epsilon_k)||_2 = 1$ for all k, the sequence $\mathbf{a}(\epsilon_k)$ contains a convergent subsequence, and let $\bar{\mathbf{a}}$ be its limit. We have

     $\mathbf{a}(\epsilon_k)^T\mathbf{z} \geq 0$ for all $\mathbf{z} \in \mathcal{S}_{-\epsilon_k}$

     and therefore $\bar{\mathbf{a}}^T\mathbf{z} \geq 0$ for all $\mathbf{z} \in interior(\mathcal{S})$, and $\bar{\mathbf{a}}^T\mathbf{z} \geq 0$ for all $\mathbf{z} \in \mathcal{S}$, which means

     $\bar{\mathbf{a}}^T\mathbf{x} \geq \bar{\mathbf{a}}^T\mathbf{y}$ for all $\mathbf{x} \in \mathcal{C}$, and $\mathbf{y} \in \mathcal{D}$.

□

Theorem 44 stated that the gradient evaluated at a point on a level set is orthogonal to the tangent hyperplane to the level set at that point. We now state the definition of a supporting hyperplane, which is special type of tangent hyperplane.

Definition 44 [Supporting Hyperplane]: *The supporting hyperplane to a set $\mathcal{C}$ at a boundary point $\mathbf{x}_0$ is defined as* $\left\{\mathbf{x}|\mathbf{a}^T\mathbf{x} = \mathbf{a}^T\mathbf{x}_0,\ \mathbf{a} \neq \mathbf{0},\ \mathbf{a}^T\mathbf{y} \leq \mathbf{a}^T\mathbf{x}_0,\ \forall\ \mathbf{y} \in \mathcal{C}\right\}$

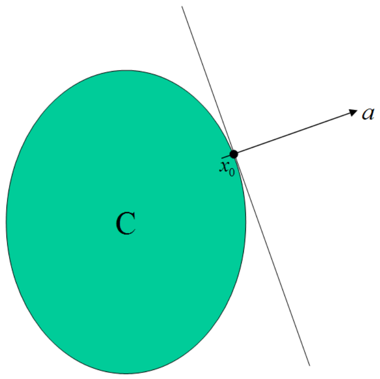


Figure 3.36: Example of a supporting hyperplane.

Figure 3.36 shows a supporting hyperplane at the point $\mathbf{x}_0$ on the boundary of a convex set $\mathcal{C}$.

For convex sets, there is an important theorem regarding supporting hyperplanes.

Theorem 53 *If the set $\mathcal{C}$ is convex, then there exists a supporting hyperplane at every boundary point of $\mathcal{C}$. As in the case of the seperating hyperplane, the supporting hyperplane need not be unique.*

Proof: If the interior of $\mathcal{C}$ is nonempty, the result follows immediately by applying the separating hyperplane theorem to the sets $\{\mathbf{x}_0\}$ and *interior*$(\mathcal{C})$. If the interior of $\mathcal{C}$ is empty, then $\mathcal{C}$ must lie in an affine set of dimension less than n, and any hyperplane containing that affine set contains $\mathcal{C}$ and $\mathbf{x}_0$, and is a (trivial) supporting hyperplane. $\square$

3.2.7 Convex Functions

Definition 45 [Convex Function]: *A function $f : \mathcal{D} \rightarrow \Re$ is convex if $\mathcal{D}$ is a convex set and*

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y}) \quad \forall\ \mathbf{x}, \mathbf{y} \in \mathcal{D} \quad 0 \leq \theta \leq 1 \tag{3.31}$$

Figure 3.37 illustrates an example convex function. A function $f : \mathcal{D} \rightarrow \Re$ is strictly convex if $\mathcal{D}$ is convex and

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) < \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})) \quad \forall\ \mathbf{x}, \mathbf{y} \in \mathcal{D} \quad 0 \leq \theta \leq 1 \tag{3.32}$$

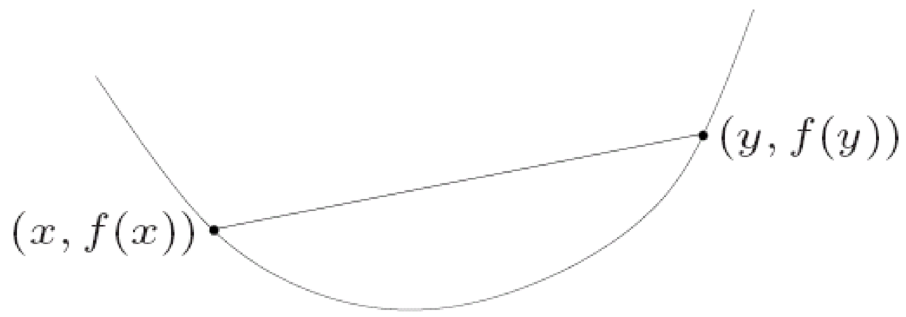


Figure 3.37: Example of convex function.

A function $f : \mathcal{D} \rightarrow \Re$ is called uniformly or strongly convex if $\mathcal{D}$ is convex and there exists a constant $c > 0$ such that

$$f(\theta\mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta)f(\mathbf{y})) - \frac{1}{2}c\theta(1-\theta)||\mathbf{x}-\mathbf{y}|| \quad \forall\ \mathbf{x},\mathbf{y} \in \mathcal{D} \quad 0 \leq \theta \leq 1 \tag{3.33}$$

A function $f : \Re^n \rightarrow \Re$ is said to be concave if the function $-f$ is convex. Examples of convex functions on the set of reals $\Re$ as well as on $\Re^n$ and $\Re^{m\times n}$ are show in Table 3.1. Examples of concave functions on the set of reals $\Re$ are show in Table 3.2. If a function is both convex and concave, it must be affine, as can be seen in the two tables.

Function type	**Domain**	**Additional Constraints**
The affine function: $ax + b$	$\Re$	Any $a, b \in \Re$
The exponential function: e^{ax}	$\Re$	Any $a \in \Re$
Powers: x^{α}	$\Re_{++}$	$\alpha \geq 1$ or $\alpha \leq 1$
Powers of absolute value: $\|x\|^p$	$\Re$	$p \geq 1$
Negative entropy: $x \log x$	$\Re_{++}$	
Affine functions of vectors: $\mathbf{a}^T\mathbf{x} + b$	$\Re^n$	
p-norms of vectors: $\|\|\mathbf{x}\|\|_p = \left(\sum_{i=1}^{n} \|x_i\|^p\right)^{1/p}$	$\Re^n$	$p \geq 1$
inf norms of vectors: $\|\|\mathbf{x}\|\|_{\infty} = \max_k \|x_k\|$	$\Re^n$	
Affine functions of matrices: $tr(A^TX) + b = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}X_{ij} + b$	$\Re^{m\times n}$	
Spectral (maximum singular value) matrix norm: $\|\|X\|\|_2 = \sigma_{max}(X) = (\lambda_{max}(X^TX))^{1/2}$	$\Re^{m\times n}$	

Table 3.1: Examples of convex functions on $\Re$, $\Re^n$ and $\Re^{m\times n}$.

3.2.8 Convexity and Global Minimum

One of the most fundamental and useful chracteristics of convex functions is that any point of local minimum point for a convex function is also a point of global minimum.

Function type	**Domain**	**Additional Constraints**
The affine function: $ax + b$	$\Re$	Any $a, b \in \Re$
Powers: x^α	$\Re_{++}$	$0 \leq \alpha \leq 1$
logarithm: $\log x$	$\Re_{++}$	

Table 3.2: Examples of concave functions on $\Re$.

Theorem 54 *Let $f : \mathcal{D} \rightarrow \Re$ be a convex function on a convex domain $\mathcal{D}$. Any point of locally minimum solution for f is also a point of its globally minimum solution.*

Proof: Suppose $\mathbf{x} \in \mathcal{D}$ is a point of local minimum and let $\mathbf{y} \in \mathcal{D}$ be a point of global minimum. Thus, $f(\mathbf{y}) < f(\mathbf{x})$. Since $\mathbf{x}$ corresponds to a local minimum, there exists an $\epsilon > 0$ such that

$$\forall\ \mathbf{z} \in \mathcal{D},\ ||\mathbf{z} - \mathbf{x}|| \leq \epsilon \Rightarrow f(\mathbf{z}) \geq f(\mathbf{x})$$

Consider a point $\mathbf{z} = \theta\mathbf{y} + (1-\theta)\mathbf{x}$ with $\theta = \frac{\epsilon}{2||\mathbf{y}-\mathbf{x}||}$. Since $\mathbf{x}$ is a point of local minimum (in a ball of radius ϵ), and since $f(\mathbf{y}) < f(\mathbf{x})$, it must be that $||\mathbf{y} - \mathbf{x}|| > \epsilon$. Thus, $0 < \theta < \frac{1}{2}$ and $\mathbf{z} \in \mathcal{D}$. Furthermore, $||\mathbf{z} - \mathbf{x}|| = \frac{\epsilon}{2}$. Since f is a convex function

$$f(\mathbf{z}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y})$$

Since $f(\mathbf{y}) < f(\mathbf{x})$, we also have

$$\theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}) < f(\mathbf{x})$$

The two equations imply that $f(\mathbf{z}) < f(\mathbf{x})$, which contradicts our assumption that $\mathbf{x}$ corresponds to a point of local minimum. That is f cannot have a point of local minimum, which does not coincide with the point $\mathbf{y}$ of global minimum. $\square$

Since any locally minimum point for a convex function also corresponds to its global minimum, we will drop the qualifiers 'locally' as well as 'globally' while referring to the points corresponding to minimum values of a convex function. For any stricly convex function, the point corresponding to the gobal minimum is also unique, as stated in the following theorem.

Theorem 55 *Let $f : \mathcal{D} \rightarrow \Re$ be a strictly convex function on a convex domain $\mathcal{D}$. Then f has a unique point corresponding to its global minimum.*

Proof: Suppose $\mathbf{x} \in \mathcal{D}$ and $\mathbf{y} \in \mathcal{D}$ with $\mathbf{y} \neq \mathbf{x}$ are two points of global minimum. That is $f(\mathbf{x}) = f(\mathbf{y})$ for $\mathbf{y} \neq \mathbf{x}$. The point $\frac{\mathbf{x}+\mathbf{y}}{2}$ also belongs to the convex set $\mathcal{D}$ and since f is strictly convex, we must have

$$f\left(\frac{\mathbf{x}+\mathbf{y}}{2}\right) < \frac{1}{2}f(\mathbf{x}) + \frac{1}{2}f(\mathbf{y}) = f(\mathbf{x})$$

which is a contradiction. Thus, the point corresponding to the minimum of f must be unique. $\square$

In the following section, we state some important properties of convex functions, including relationships between convex functions and convex sets, and first and second order conditions for convexity. We will also draw relationships between the definitions of convexity and strict convexity stated here, with the definitions on page 168 for the single variable case.

3.2.9 Properties of Convex Functions

We will first extend the domain of a convex function to all $\Re^n$, while retaining its convexity and preserving its value in the domain.

Definition 46 [Extended value extension]: *If $f : \mathcal{D} \rightarrow \Re$, with $\mathcal{D} \subseteq \Re^n$ is a convex function, then we define its extended-valued extension $\widetilde{f} : \Re^n \rightarrow \Re$ as*

$$\widetilde{f}(\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{D} \\ \infty & \text{if } \mathbf{x} \notin \mathcal{D} \end{cases} \tag{3.34}$$

In what follows, we will assume if necessary, that all convex functions are implicitly extended to the domain $\Re^n$. A useful technique for verifying the convexity of a function is to investigate its convexity, by restricting the function to a line and checking for the convexity of a function of single variable. This technique is hinged on the following theorem.

Theorem 56 *A function $f : \mathcal{D} \rightarrow \Re$ is (strictly) convex if and only if the function $\phi : \mathcal{D}_\phi \rightarrow \Re$ defined below, is (strictly) convex in t for every $\mathbf{x} \in \Re^n$ and for every $\mathbf{h} \in \Re^n$*

$$\phi(t) = f(\mathbf{x} + t\mathbf{h})$$

with the domain of ϕ given by $\mathcal{D}_\phi = \{t | \mathbf{x} + t\mathbf{h} \in \mathcal{D}\}$.

Proof: We will prove the necessity and sufficiency of the convexity of ϕ for a convex function f. The proof for necessity and sufficiency of the strict convexity of ϕ for a strictly convex f is very similar and is left as an exercise.

Proof of Necessity: Assume that f is convex. And we need to prove that $\phi(t) = f(\mathbf{x} + t\mathbf{h})$ is also convex. Let $t_1, t_2 \in \mathcal{D}_\phi$ and $\theta \in [0, 1]$. Then,

$$\begin{aligned} \phi(\theta t_1 + (1-\theta)t_2) &= f\left(\theta(\mathbf{x} + t_1\mathbf{h}) + (1-\theta)(\mathbf{x} + t_2\mathbf{h})\right) \\ &\leq \theta f\left((\mathbf{x} + t_1\mathbf{h})\right) + (1-\theta) f\left((\mathbf{x} + t_2\mathbf{h})\right) = \theta\phi(t_1) + (1-\theta)\phi(t_2) \end{aligned} \tag{3.35}$$

Thus, ϕ is convex.

Proof of Sufficiency: Assume that for every $\mathbf{h} \in \Re^n$ and every $\mathbf{x} \in \Re^n$, $\phi(t) = f(\mathbf{x} + t\mathbf{h})$ is convex. We will prove that f is convex. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$. Take, $\mathbf{x} = \mathbf{x}_1$ and $\mathbf{h} = \mathbf{x}_2 - \mathbf{x}_1$. We know that $\phi(t) = f\left(\mathbf{x}_1 + t(\mathbf{x}_2 - \mathbf{x}_1)\right)$ is convex, with $\phi(1) = f(\mathbf{x}_2)$ and $\phi(0) = f(\mathbf{x}_1)$. Therefore, for any $\theta \in [0, 1]$

$$\begin{aligned} f(\theta\mathbf{x}_2 + (1-\theta)\mathbf{x}_1) &= \phi(\theta) \\ \leq \theta\phi(1) + (1-\theta)\phi(0) &\leq \theta f(\mathbf{x}_2) + (1-\theta)f(\mathbf{x}_1) \end{aligned} \tag{3.36}$$

This implies that f is convex. □

Next, we will draw the parallel between convex sets and convex functions by introducing the concept of the *epigraph* of a function.

Definition 47 [Epigraph]: *Let $\mathcal{D} \subseteq \Re^n$ be a nonempty set and $f : \mathcal{D} \to \Re$. The set $\{(\mathbf{x}, f(\mathbf{x})|\mathbf{x} \in \mathcal{D}\}$ is called graph of f and lies in $\Re^{n+1}$. The epigraph of f is a subset of $\Re^{n+1}$ and is defined as*

$$epi(f) = \{(\mathbf{x}, \alpha)|f(\mathbf{x}) \leq \alpha,\ \mathbf{x} \in \mathcal{D},\ \alpha \in \Re\} \tag{3.37}$$

In some sense, the epigraph is the set of points lying above the graph of f. Similarly, the hypograph of f is a subset of $\Re^{n+1}$, lying above the graph of f and is defined by

$$hyp(f) = \{(\mathbf{x}, \alpha)|f(\mathbf{x}) \geq \alpha,\ \mathbf{x} \in \mathcal{D},\ \alpha \in \Re\} \tag{3.38}$$

There is a one to one correspondence between the convexity of function f and that of the set $epi(f)$, as stated in the following theorem.

Theorem 57 *Let $\mathcal{D} \subseteq \Re^n$ be a nonempty convex set, and $f : \mathcal{D} \to \Re$. Then f is convex if and only if $epi(f)$ is a convex set.*

Proof: Let f be convex. For any $(\mathbf{x}_1, \alpha_1) \in epi(f)$ and $(\mathbf{x}_2, \alpha_2) \in epi(f)$ and any $\theta \in (0, 1)$,

$$f(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta)f(\mathbf{x}_2)) \leq \theta\alpha_1 + (1-\theta)\alpha_2$$

Since $\mathcal{D}$ is convex, $\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \mathcal{D}$. Therefore, $(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2, \theta\alpha_1 + (1-\theta)\alpha_2) \in epi(f)$. Thus, $epi(f)$ is convex if f is convex. This proves the necessity part.

To prove sufficiency, assume that $epi(f)$ is convex. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$. So, $(\mathbf{x}_1, f(\mathbf{x}_1)) \in epi(f)$ and $(\mathbf{x}_2, f(\mathbf{x}_2)) \in epi(f)$. Since $epi(f)$ is convex, for $\theta \in (0, 1)$,

$$(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2, \theta\alpha_1 + (1-\theta)\alpha_2) \in epi(f)$$

which implies that $f(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2) \leq \theta f(\mathbf{x}_1) + (1-\theta)f(\mathbf{x}_2))$ for any $\theta \in (0, 1)$. This proves the sufficiency. □

There is also a correspondence between the convexity of a function and the convexity of its *sublevel sets*.

Definition 48 [Sublevel Sets]: *Let $\mathcal{D} \subseteq \Re^n$ be a nonempty set and $f : \mathcal{D} \rightarrow \Re$. The set*

$$L_\alpha(f) = \{\mathbf{x} | \mathbf{x} \in \mathcal{D},\ f(\mathbf{x}) \leq \alpha\}$$

is called the $\alpha-$sub-level set of f.

The correspondence between the convexity of f and its $\alpha-$sub-level set is stated in the following theorem. Unlike the correspondence with the epigraph, the correspondence with the $\alpha-$sub-level set is not one to one.

Theorem 58 *Let $\mathcal{D} \subseteq \Re^n$ be a nonempty convex set, and $f : \mathcal{D} \rightarrow \Re$ be a convex function. Then $L_\alpha(f)$ is a convex set for any $\alpha \in \Re$.*

Proof: Consider $\mathbf{x}_1, \mathbf{x}_2 \in L_\alpha(f)$. Then by definition of the level set, $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$, $f(\mathbf{x}_1) \leq \alpha$ and $f(\mathbf{x}_2) \leq \alpha$. From convexity of $\mathcal{D}$ it follows that for all $\theta \in (0, 1)$, $\mathbf{x} = \theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2 \in \mathcal{D}$. Moreover, since f is also convex,

$$f(\mathbf{x}) \leq \theta f(\mathbf{x}_1) + (1-\theta) f(\mathbf{x}_2) \leq \theta\alpha + (1-\theta)\alpha = \alpha$$

which implies that $\mathbf{x} \in L_\alpha(f)$. Thus, $L_\alpha(f)$ is a convex set. $\square$The converse of this theorem does not hold. To illustrate this, consider the function $f(\mathbf{x}) = \frac{x_2}{1+2x_1^2}$. The 0-sublevel set of this function is $\{(x_1, x_2) \mid x_2 \leq 0\}$, which is convex. However, the function $f(\mathbf{x})$ itself is not convex.

An important property of a convex function is that it is continuous in the interior of its domain.

Theorem 59 *Let $f : \mathcal{D} \rightarrow \Re$ be a convex function with $\mathcal{D} \subseteq \Re^n$ being a convex set. Let $\mathcal{S} \subset \mathcal{D}$ be an open convex set. Then f is continuous on $\mathcal{S}$.*

Proof: Let us consider a point $\mathbf{x}_0 \in \mathcal{S}$. Since $\mathcal{S}$ is an open convex set, we can find $n+1$ points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1}$ such that the interior of the convex hull

$$\mathcal{C} = \left\{ \mathbf{x} | \mathbf{x} = \sum_{i=1}^{n+1} a_i \mathbf{x}_i,\ a_i \geq 0,\ \sum_{1}^{n+1} a_i = 1 \right\}$$

is not empty and $\mathbf{x}_0 \in interior(\mathcal{C})$. Let $M = \max\limits_{1 \leq i \leq n+1} f(x_i)$. Then, for any $\mathbf{x} = \sum\limits_{i=1}^{n+1} a_i \mathbf{x}_i \in \mathcal{C}$,

$$f(\mathbf{x}) = f\left(\sum_{i=1}^{n+1} a_i \mathbf{x}_i\right) \leq \sum_{i=1}^{n+1} a_i f(\mathbf{x}_i) \leq M$$

Since $\mathbf{x}_0 \in \mathcal{C}$, there exists a $\delta > 0$ such that $B(\mathbf{x}_0, \delta) \subset \mathcal{C}$, where, $B(\mathbf{x}_0, \delta) = \{\mathbf{x} | ||\mathbf{x} - \mathbf{x}_0|| \leq \delta\}$. Therefore, $\mathbf{x}_0$ can be expressed as a convex combination of $(\mathbf{x}_0 + \theta\mathbf{h}$ and $\mathbf{x}_0 - \mathbf{h}$ for some $\mathbf{h} \in B(\mathbf{x}_0, \delta)$ and some $\theta \in [0, 1]$.

$$\mathbf{x}_0 = \frac{1}{1+\theta}(\mathbf{x}_0 + \theta\mathbf{h}) + \frac{\theta}{1+\theta}(\mathbf{x}_0 - \mathbf{h})$$

Since f is convex on $\mathcal{C}$,

$$f(\mathbf{x}_0) \leq \frac{1}{1+\theta} f(\mathbf{x}_0 + \theta\mathbf{h}) + \frac{\theta}{1+\theta} f(\mathbf{x}_0 - \mathbf{h})$$

From this, we can conclude that

$$f(\mathbf{x}_0 + \theta\mathbf{h}) - f(\mathbf{x}_0) \geq \theta \left(f(\mathbf{x}_0 - f(\mathbf{x}_0 - \mathbf{h}) \right) \geq -\theta \left(M - f(\mathbf{x}_0) \right) \tag{3.39}$$

On the other hand,

$$f(\mathbf{x}_0 + \theta\mathbf{h}) \leq \theta f(\mathbf{x}_0 + \mathbf{h}) + (1-\theta) f(\mathbf{x}_0)$$

which implies that

$$f(\mathbf{x}_0 + \theta\mathbf{h}) - f(\mathbf{x}_0) \leq \theta \left(f(\mathbf{x}_0 + \mathbf{h}) - f(\mathbf{x}_0) \leq \theta \left(M - f(\mathbf{x}_0) \right) \right. \tag{3.40}$$

From equations 3.39 and 3.40, we can infer that

$$|f(\mathbf{x}_0 + \theta\mathbf{h}) - f(\mathbf{x}_0)| \leq \theta |f(\mathbf{x}_0) - M|$$

For a given $\epsilon > 0$, select $\delta' \leq \delta$ such that $\delta'|f(\mathbf{x}_0) - M| \leq \epsilon\delta$. Then $\mathbf{d} = \theta\mathbf{h}$ with $||\mathbf{h}|| = \delta$, implies that $d \in B(\mathbf{x}_0, \delta)$ and $f(\mathbf{x}_0 + \mathbf{d}) - f(\mathbf{x}_0)| \leq \epsilon$. This proves the theorem. $\Box$

Analogous to the definition of increasing functions introduced on page number 164, we next introduce the concept of monotonic functions. This concept is very useful for characterization of a convex function.

Definition 49 *Let $\mathbf{f} : \mathcal{D} \rightarrow \Re^n$ and $\mathcal{D} \subseteq \Re^n$. Then*

1. *$\mathbf{f}$ is monotone on $\mathcal{D}$ if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$,*

$$\left(\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2) \right)^T (\mathbf{x}_1 - \mathbf{x}_2) \geq 0 \tag{3.41}$$

2. *$\mathbf{f}$ is strictly monotone on $\mathcal{D}$ if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$ with $\mathbf{x}_1 \neq \mathbf{x}_2$,*

$$\left(\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2) \right)^T (\mathbf{x}_1 - \mathbf{x}_2) > 0 \tag{3.42}$$

3. *$\mathbf{f}$ is uniformly or strongly monotone on $\mathcal{D}$ if for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$, there is a constant $c > 0$ such that*

$$\left(\mathbf{f}(\mathbf{x}_1) - \mathbf{f}(\mathbf{x}_2) \right)^T (\mathbf{x}_1 - \mathbf{x}_2) \geq c||\mathbf{x}_1 - \mathbf{x}_2||^2 \tag{3.43}$$

First-Order Convexity Conditions

The first order convexity condition for differentiable functions is provided by the following theorem:

Theorem 60 *Let $f : \mathcal{D} \rightarrow \Re$ be a differentiable convex function on an open convex set $\mathcal{D}$. Then:*

1. *f is convex if and only if, for any $\mathbf{x}, \mathbf{y} \in \mathcal{D}$,*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \tag{3.44}$$

2. *f is strictly convex on $\mathcal{D}$ if and only if, for any $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, with $\mathbf{x} \neq \mathbf{y}$,*

$$f(\mathbf{y}) > f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \tag{3.45}$$

3. *f is strongly convex on $\mathcal{D}$ if and only if, for any $\mathbf{x}, \mathbf{y} \in \mathcal{D}$,*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{1}{2}c||\mathbf{y} - \mathbf{x}||^2 \tag{3.46}$$

for some constant $c > 0$.

Proof:

Sufficiency: The proof of sufficiency is very similar for all the three statements of the theorem. So we will prove only for statement (3.44). Suppose (3.44) holds. Consider $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$ and any $\theta \in (0, 1)$. Let $\mathbf{x} = \theta\mathbf{x}_1 + (1 - \theta)\mathbf{x}_2$. Then,

$$\begin{aligned} f(\mathbf{x}_1) &\geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{x}_1 - \mathbf{x}) \\ f(\mathbf{x}_2) &\geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{x}_2 - \mathbf{x}) \end{aligned} \tag{3.47}$$

Adding $(1 - \theta)$ times the second inequality to θ times the first, we get,

$$\theta f(\mathbf{x}_1) + (1 - \theta) f(\mathbf{x}_2) \geq f(\mathbf{x})$$

which proves that $f(\mathbf{x})$ is a convex function. In the case of strict convexity, strict inequality holds in (3.47) and it follows through. In the case of strong convexity, we need to additionally prove that

$$\theta\frac{1}{2}c||\mathbf{x} - \mathbf{x}_1||^2 + (1 - \theta)\frac{1}{2}c||\mathbf{x} - \mathbf{x}_2||^2 = \frac{1}{2}c\theta(1 - \theta)||\mathbf{x}_2 - \mathbf{x}_1||^2$$

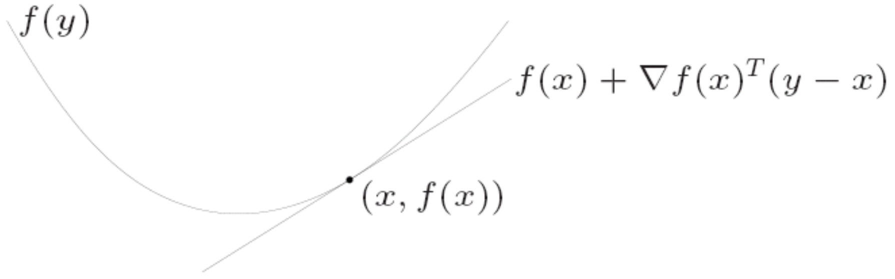


Figure 3.38: Figure illustrating Theorem 60.

Necessity: Suppose f is convex. Then for all $\theta \in (0,1)$ and $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{D}$, we must have

$$f(\theta\mathbf{x}_2 + (1-\theta)\mathbf{x}_1) \leq \theta f(\mathbf{x}_2) + (1-\theta)f(\mathbf{x}_1)$$

Thus,

$$\nabla^T f(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) = \lim_{\theta \to 0} \frac{f\left(\mathbf{x}_1 + \theta(\mathbf{x}_2 - \mathbf{x}_1)\right) - f\left(\mathbf{x}_1\right)}{\theta} \leq f(\mathbf{x}_2) - f(\mathbf{x}_1)$$

This proves necessity for (3.44). The necessity proofs for (3.45) and (3.46) are very similar, except for a small difference for the case of strict convexity; the strict inequality is not preserved when we take limits. Suppose equality does hold in the case of strict convexity, that is for a strictly convex function f, let

$$f(\mathbf{x}_2) = f(\mathbf{x}_1) + \nabla^T f(\mathbf{x}_1)(\mathbf{x}_2 - \mathbf{x}_1) \tag{3.48}$$

for some $\mathbf{x}_2 \neq \mathbf{x}_1$. Because f is stricly convex, for any $\theta \in (0,1)$ we can write

$$f\left(\theta\mathbf{x}_1 + (1-\theta)\mathbf{x}_2\right) = f\left(\mathbf{x}_2 + \theta(\mathbf{x}_1 - \mathbf{x}_2)\right) < \theta f(\mathbf{x}_1) + (1-\theta)f(\mathbf{x}_2) \tag{3.49}$$

Since (3.44) is already proved for convex functions, we use it in conjunction with (3.48), and (3.49), to get

$$f(\mathbf{x}_2) + \theta\nabla^T f(\mathbf{x}_2)(\mathbf{x}_1 - \mathbf{x}_2) \leq f\left(\mathbf{x}_2 + \theta(\mathbf{x}_1 - \mathbf{x}_2)\right) < f(\mathbf{x}_2) + \theta\nabla^T f(\mathbf{x}_2)(\mathbf{x}_1 - \mathbf{x}_2)$$

which is a contradiction. Thus, equality can never hold in (3.44) for any $\mathbf{x}_1 \neq \mathbf{x}_2$. This proves the necessity of (3.45). □

The geometrical interpretation of theorem 60 is that at any point, the linear approximation based on a local derivative gives a lower estimate of the function, *i.e.* the convex function always lies above the supporting hyperplane at that point. This is pictorially depicted in Figure 3.38. There are some implications of theorem 60 for strongly convex functions. We state them next.

Definition 50 [Some corollaries of theorem 60 for strongly convex functions]: *For a fixed $\mathbf{x}$, the right hand side of the inequality (3.46) is a convex quadratic function of $\mathbf{y}$. Thus, the critical point of the RHS should correspond to the minimum value that the RHS could take. This yields another lower bound on $f(\mathbf{y})$.*

$$f(\mathbf{y}) \geq f(\mathbf{x}) - \frac{1}{2c}||\nabla f(\mathbf{x})||_2^2 \tag{3.50}$$

Since this holds for any $\mathbf{y} \in \mathcal{D}$, we have

$$\min_{\mathbf{y}\in\mathcal{D}} f(\mathbf{y}) \geq f(\mathbf{x}) - \frac{1}{2c}||\nabla f(\mathbf{x})||_2^2 \tag{3.51}$$

which can be used to bound the suboptimality of a point $\mathbf{x}$ in terms of $||\nabla f(\mathbf{x})||_2$. This bound comes handy in theoretically understanding the convergence of gradient methods. If $\widehat{\mathbf{y}} = \min\limits_{\mathbf{y}\in\mathcal{D}} f(\mathbf{y})$, we can also derive a bound on the distance between any point $\mathbf{x} \in \mathcal{D}$ and the point of optimality $\widehat{\mathbf{y}}$.

$$||\mathbf{x} - \widehat{\mathbf{y}}||_2 \leq \frac{2}{c}||\nabla f(\mathbf{x})||_2 \tag{3.52}$$

Theorem 60 motivates the definition of the *subgradient* for non-differentiable convex functions, which has properties very similar to the gradient vector.

Definition 51 [Subgradient]: *Let $f : \mathcal{D} \rightarrow \Re$ be a convex function defined on a convex set $\mathcal{D}$. A vector $\mathbf{h} \in \Re^n$ is said to be a subgradient of f at the point $\mathbf{x} \in \mathcal{D}$ if*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{h}^T(\mathbf{y} - \mathbf{x})$$

for all $\mathbf{y} \in \mathcal{D}$. The set of all such vectors is called the subdifferential of f at $\mathbf{x}$.

For a differentiable convex function, the gradient at point $\mathbf{x}$ is the only subgradient at that point. Most properties of differentiable convex functions that hold in terms of the gradient also hold in terms of the subgradient for non-differentiable convex functions. Theorem 60 gives a very simple optimality criterion for a differentiable function f.

Theorem 61 *Let $f : \mathcal{D} \rightarrow \Re$ be a convex function defined on a convex set $\mathcal{D}$. A point $\mathbf{x} \in \mathcal{D}$ corresponds to a minimum if and only if*

$$\nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \geq 0$$

for all $\mathbf{y} \in \mathcal{D}$.

If $\nabla f(\mathbf{x})$ is nonzero, it defines a supporting hyperplane to $\mathcal{D}$ at the point $\mathbf{x}$. Theorem 62 implies that for a differentiable convex function defined on an open set, every critical point must be a point of (global) minimum.

Theorem 62 *Let $f : \mathcal{D} \rightarrow \Re$ be differentiable and convex on an open convex domain $\mathcal{D} \subseteq \Re^n$. Then $\mathbf{x}$ is a critical point of f if and only if it is a (global) minimum.*

Proof: If $\mathbf{x}$ is a global minimum, it is a local minimum and by theorem 45, it must be a critical point and therefore $\nabla f(\mathbf{x}) = 0$. Conversely, let $\nabla f(\mathbf{x}) = 0$, By theorem 60, we know that for all $\mathbf{y} \in \mathcal{D}$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x})$$

Substituting $\nabla f(\mathbf{x}) = 0$ in this inequality, we get for all $\mathbf{y} \in \mathcal{D}$,

$$f(\mathbf{y}) \geq f(\mathbf{x})$$

That is, $\mathbf{x}$ corresponds to a (global) minimum. □

Based on the definition of monotonic functions in definition 49, we show the relationship between convexity of a function and monotonicity of its gradient in the next theorem.

Theorem 63 *Let $f : \mathcal{D} \rightarrow \Re$ with $\mathcal{D} \subseteq \Re^n$ be differentiable on the convex set $\mathcal{D}$. Then,*

1. *f is convex on $\mathcal{D}$ if and only if is its gradient ∇f is monotone. That is, for all $\mathbf{x}, \mathbf{y} \in \Re$*

$$\left(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\right)^T (\mathbf{x} - \mathbf{y}) \geq 0 \tag{3.53}$$

2. *f is strictly convex on $\mathcal{D}$ if and only if is its gradient ∇f is strictly monotone. That is, for all $\mathbf{x}, \mathbf{y} \in \Re$ with $\mathbf{x} \neq \mathbf{y}$,*

$$\left(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\right)^T (\mathbf{x} - \mathbf{y}) > 0 \tag{3.54}$$

3. *f is uniformly or strongly convex on $\mathcal{D}$ if and only if is its gradient ∇f is uniformly monotone. That is, for all $\mathbf{x}, \mathbf{y} \in \Re$,*

$$\left(\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\right)^T (\mathbf{x} - \mathbf{y}) \geq c||\mathbf{x} - \mathbf{y}||^2 \tag{3.55}$$

for some constant $c > 0$.

Proof:

Necessity: Suppose f is uniformly convex on $\mathcal{D}$. Then from theorem 60, we know that for any $\mathbf{x}, \mathbf{y} \in \mathcal{D}$,

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) - \frac{1}{2}c||\mathbf{y} + \mathbf{x}||^2$$
$$f(\mathbf{x}) \geq f(\mathbf{y}) + \nabla^T f(\mathbf{y})(\mathbf{x} - \mathbf{y}) - \frac{1}{2}c||\mathbf{x} + \mathbf{y}||^2$$

Adding the two inequalities, we get (3.55). If f is convex, the inequalities hold with $c = 0$, yielding (3.54). If f is strictly convex, the inequalities will be strict, yielding (3.54).

Sufficiency: Suppose ∇f is monotone. For any fixed $\mathbf{x}, \mathbf{y} \in \mathcal{D}$, consider the function $\phi(t) = f(\mathbf{x} + t(\mathbf{y} - \mathbf{x}))$. By the mean value theorem applied to $\phi(t)$, we should have for some $t \in (0, 1)$,

$$\phi(1) - \phi(0) = \phi'(t) \tag{3.56}$$

Letting $\mathbf{z} = \mathbf{x} + t(\mathbf{y} - \mathbf{x})$, (3.56) translates to

$$f(\mathbf{y}) - f(\mathbf{x}) = \nabla^T f(\mathbf{z})(\mathbf{y} - \mathbf{x}) \tag{3.57}$$

Also, by definition of monotonicity of ∇f, (from (3.53)),

$$(\nabla f(\mathbf{z}) - \nabla f(\mathbf{x}))^T (\mathbf{y} - \mathbf{x}) = \frac{1}{t}(\nabla f(\mathbf{z}) - \nabla f(\mathbf{x}))^T (\mathbf{z} - \mathbf{x}) \geq 0 \tag{3.58}$$

Combining (3.57) with (3.58), we get,

$$\begin{aligned} f(\mathbf{y}) - f(\mathbf{x}) &= (\nabla f(\mathbf{z}) - f(\mathbf{x}))^T (\mathbf{y} - \mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \\ &\geq \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \end{aligned} \tag{3.59}$$

By theorem 60, this inequality proves that f is convex. Strict convexity can be similarly proved by using the strict inequality in (3.58) inherited from strict monotonicity, and letting the strict inequality follow through to (3.59). For the case of strong convexity, from (3.55), we have

$$\begin{aligned} \phi'(t) - \phi'(0) &= (\nabla f(\mathbf{z}) - f(\mathbf{x}))^T (\mathbf{y} - \mathbf{x}) \\ = \frac{1}{t}(\nabla f(\mathbf{z}) - f(\mathbf{x}))^T (\mathbf{z} - \mathbf{x}) &\geq \frac{1}{t}c||\mathbf{z} - \mathbf{x}||^2 = ct||\mathbf{y} - \mathbf{x}||^2 \end{aligned} \tag{3.60}$$

Therefore,

$$\phi(1) - \phi(0) - \phi'(0) = \int_0^1 [\phi'(t) - \phi'(0)]dt \geq \frac{1}{2}c||\mathbf{y} - \mathbf{x}||^2 \qquad (3.61)$$

which translates to

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla^T f(\mathbf{x})(\mathbf{y} - \mathbf{x}) + \frac{1}{2}c||\mathbf{y} - \mathbf{x}||^2$$

By theorem 60, f must be strongly convex. $\Box$

Second Order Condition

For twice continuously differentiable convex functions the convexity condition can be characterized as follows.

Theorem 64 *A twice differential function $f : \mathcal{D} \rightarrow \Re$ for a nonempty open convex set $\mathcal{D}$*

1. *is convex if and only if its domain is convex and its Hessian matrix is positive semidefinite at each point in $\mathcal{D}$. That is*

$$\nabla^2 f(\mathbf{x}) \succeq 0 \quad \forall\ \mathbf{x} \in \mathcal{D} \qquad (3.62)$$

2. *is strictly convex if its domain is convex and its Hessian matrix is positive definite at each point in $\mathcal{D}$. That is*

$$\nabla^2 f(\mathbf{x}) \succ 0 \quad \forall\ \mathbf{x} \in \mathcal{D} \qquad (3.63)$$

3. *is uniformly convex if and only if its domain is convex and its Hessian matrix is uniformly positive definite at each point in $\mathcal{D}$. That is, for any $\mathbf{v} \in \Re^n$ and any $\mathbf{x} \in \mathcal{D}$, there exists a $c > 0$ such that*

$$\mathbf{v}^T \nabla^2 f(\mathbf{x})\mathbf{v} \geq c||\mathbf{v}||^2 \qquad (3.64)$$

In other words

$$\nabla^2 f(\mathbf{x}) \succeq cI_{n\times n}$$

where $I_{n\times n}$ is the $n \times n$ identity matrix and $\succeq$ corresponds to the positive semidefinite inequality. That is, the function f is strongly convex iff $\nabla^2 f(\mathbf{x}) - cI_{n\times n}$ is positive semidefinite, for all $\mathbf{x} \in \mathcal{D}$ and for some constant $c > 0$, which corresponds to the positive minimum curvature of f.

Proof: We will prove only the first statement in the theorem; the other two statements are proved in a similar manner.

Necessity: Suppose f is a convex function, and consider a point $\mathbf{x} \in \mathcal{D}$. We will prove that for any $\mathbf{h} \in \Re^n$, $\mathbf{h}^T \nabla^2 f(\mathbf{x})\mathbf{h} \geq 0$. Since f is convex, by theorem 60, we have

$$f(\mathbf{x} + t\mathbf{h}) \geq f(\mathbf{x}) + t\nabla^T f(\mathbf{x})\mathbf{h} \tag{3.65}$$

Consider the function $\phi(t) = f(\mathbf{x} + t\mathbf{h})$ considered in theorem 56, defined on the domain $\mathcal{D}_\phi = [0, 1]$. Using the chain rule,

$$\phi'(t) = \sum_{i=1}^{n} f_{x_i}(\mathbf{x} + t\mathbf{h})\frac{dx_i}{dt} = \mathbf{h}^T.\nabla f(\mathbf{x} + t\mathbf{h})$$

Since f has partial and mixed partial derivatives, ϕ' is a differentiable function of t on $\mathcal{D}_\phi$ and

$$\phi''(t) = \mathbf{h}^T \nabla^2 f(\mathbf{x} + t\mathbf{h})\mathbf{h}$$

Since ϕ and ϕ' are continous on $\mathcal{D}_\phi$ and ϕ' is differentiable on $int(\mathcal{D}_\phi)$, we can make use of the Taylor's theorem (30) with $n = 3$ to obtain:

$$\phi(t) = \phi(0) + t.\phi'(0) + t^2.\frac{1}{2}\phi''(0) + O(t^3)$$

Writing this equation in terms of f gives

$$f(\mathbf{x} + t\mathbf{h}) = f(\mathbf{x}) + t\mathbf{h}^T \nabla f(\mathbf{x}) + t^2\frac{1}{2}h^T \nabla^2 f(\mathbf{x})\mathbf{h} + O(t^3)$$

In conjunction with (3.65), the above equation implies that

$$\frac{t^2}{2}h^T \nabla^2 f(\mathbf{x})\mathbf{h} + O(t^3) \geq 0$$

Dividing by t^2 and taking limits as $t \rightarrow 0$, we get

$$h^T \nabla^2 f(\mathbf{x})\mathbf{h} \geq 0$$

Sufficiency: Suppose that the Hessian matrix is positive semidefinite at each point $\mathbf{x} \in \mathcal{D}$. Consider the same function $\phi(t)$ defined above with $\mathbf{h} = \mathbf{y} - \mathbf{x}$ for $\mathbf{y}, \mathbf{x} \in \mathcal{D}$. Applying Taylor's theorem (30) with $n = 2$ and $a = 0$, we obtain,

$$\phi(1) = \phi(0) + t.\phi'(0) + t^2.\frac{1}{2}\phi''(c)$$

for some $c \in (0, 1)$. Writing this equation in terms of f gives

$$f(\mathbf{x}) = f(\mathbf{y}) + (\mathbf{x} - \mathbf{y})^T \nabla f(\mathbf{y}) + \frac{1}{2}(\mathbf{x} - \mathbf{y})^T \nabla^2 f(\mathbf{z})(\mathbf{x} - \mathbf{y})$$

where $\mathbf{z} = \mathbf{y} + c(\mathbf{x} - \mathbf{y})$. Since $\mathcal{D}$ is convex, $\mathbf{z} \in \mathcal{D}$. Thus, $\nabla^2 f(\mathbf{z}) \succeq 0$. It follows that

$$f(\mathbf{x}) \geq f(\mathbf{y}) + (\mathbf{x} - \mathbf{y})^T \nabla f(\mathbf{y})$$

By theorem 60, the function f is convex. $\square$

Examples of differentiable/twice differentiable convex functions, along with the value of their respective gradients/hessians are tabulated in Table 3.3.

Function type	**Constraints**	**Gradient/Hessian**
Quadratic : $\frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + c$	$A \succeq 0$	$\nabla^2 f(\mathbf{x}) = P$
Quadratic over linear: $\frac{x^2}{y} \geq 0$	$y > 0$	$\nabla^2 f(x,y) = \frac{2}{y^3}\begin{bmatrix} y^2 & -xy \\ -xy & x^2 \end{bmatrix}$
Log-sum-exp: $\log \sum_{k=1}^{n} exp(x_k)$		$\nabla^2 f(x) = \frac{1}{(\mathbf{1}^T\mathbf{z})^2}\left((\mathbf{1}^T\mathbf{z})\ \boldsymbol{diag}(\mathbf{z}) - \mathbf{z}\mathbf{z}^T\right)$ where $\mathbf{z} = [e^{x_1}, e^{x_1}, \ldots, e^{x_n}]$
Negative Geometric mean: $-\left(\prod_{k=1}^{n} x_k\right)^{\frac{1}{n}}$	$\mathbf{x} \in \Re^n_{++}$	$\nabla^2 f(x) = \frac{\prod_{i=1} n x_i^{1/n}}{n^2}\left(n\ \boldsymbol{diag}(\frac{1}{x_1^2}, \ldots, \frac{1}{x_n^2}) - qq^T\right)$

Table 3.3: Examples of twice differentiable convex functions on $\Re$.

3.2.10 Convexity Preserving Operations on Functions

In practice if you want to establish the convexity of a function f, you could either

1. Prove it from first principles, i.e., using the definition of convexity or
2. If f is twice differentiable, show that $\nabla^2 f(x) \succeq 0$
3. Show that f is obtained from simple convex functions by operations that preserve complexity. Following are operations on functions that preserve complexity (proofs omitted, since they are trivial):

   - **Nonnegative weighted sum:** $f = \sum_{i=1}^{n} \alpha_i f_i$ is convex if each f_i for $1 \leq i \leq n$ is convex and $\alpha_i \geq 0, 1 \leq i \leq n$.
   - **Composition with affine function:** $f(Ax + b)$ is convex if f is convex. For example:
     - The log barrier for linear inequalities, $f(x) = -\sum_{i=1}^{m} \log(b_i - a_i^T x)$, is convex since $-\log(x)$ is convex.
     - Any norm of an affine function, $f(x) = ||Ax + b||$, is convex.
   - **Pointwise maximum:** If $f_1, f_2, \ldots, f_m$ are convex, then $f(x) = max\{f_1(x), f_2(x), \ldots, f_m(x)\}$ is also convex, For example:

  - Sum of r largest components of $\mathbf{x} \in \Re^n$ $f(\mathbf{x}) = x_{[1]} + x_{[2]} + \ldots + x_{[r]}$, where $x_{[1]}$ is the i^{th} largest component of $\mathbf{x}$, is a convex function.

- **Pointwise supremum:** If $f(x, y)$ is convex in x for every $y \in \mathcal{S}$, then $g(x) = \sup\limits_{y \in \mathcal{S}} f(x, y)$ is convex. For example:
  - The function that returns the maximum eigenvalue of a symmetric matrix X, *viz.*, $\lambda_{max}(X) = \sup\limits_{y \in \mathcal{S}} f(x, y)$ is a convex function of the symmetrix matrix X.

- **Composition with functions:** Let $h : \Re^k \rightarrow \Re$ with $h(x) = \infty, \forall\ x \notin \mathbf{d}om\ h$ and $g : \Re^n \rightarrow \Re^k$. Define $f(x) = h(g(x))$. f is convex if
  - g_i is convex, h is convex and nondecreasing in each argument
  - or g_i is concave, h is convex and nonincreasing in each argument

  Some examples illustrating this property are:
  - $exp\ g(x)$ is convex if g is convex
  - $\sum\limits_{i=1}^{m} \log g_i(x)$ is concave if g_i are concave and positive
  - $\log \sum\limits_{i=1}^{m} \exp g_i(x)$ is convex if g_i are convex
  - $1/g(x)$ is convex if g is concave and positive

- **Infimum:** If $f(x, y)$ is convex in (x, y) and $\mathcal{C}$ is a convex set, then $g(x) = \inf\limits_{y \in \mathcal{C}} f(x, y)$ is convex. For example:
  - Let $f(x, \mathcal{S})$ that returns the distance of a point x to a convex set $\mathcal{S}$. That is $f(x, \mathcal{S}) = \inf\limits_{y \in \mathcal{S}} ||x - y||$. Then $f(x, \mathcal{S})$ is a convex.

- **Perspective Function:** The perspective of a function $f : \Re^n \rightarrow \Re$ is the function $g : R^n \times \Re \rightarrow \Re$, $g(x, t) = tf(x/t)$. Function g is convex if f is convex on $\mathbf{d}omg = \{(x, t) | x/t \in \mathbf{d}omf, t > 0\}$. For example,
  - The perspective of $f(x) = x^T x$ is (quadratic-over-linear) function $g(x, t) = \frac{x^T x}{t}$ and is convex.
  - The perspective of negative logarithm $f(x) = -\log x$ is the relative entropy function $g(x, t) = t \log t - t \log x$ and is convex.

3.3 Convex Optimization Problem

Formally, a convex program is defined as

$$\min_{\mathbf{x} \in \mathcal{X}} c^T x \tag{3.66}$$

where $\mathcal{X} \subset \Re^n$ is a convex set and $\mathbf{x}$ is a vector of n optimization or decision variables. In applications, convex optimization programs usually arise in the form:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & A\mathbf{x} = \mathbf{b} \\ \text{variable } \mathbf{x} = (x_1, \ldots, x_n) & \end{array} \tag{3.67}$$

If it is given that the functions $f, g_1, \ldots, g_m$ are convex, by theorem 58, the feasible set $\mathcal{X}$ of this problem, which is the intersection of a finite number of $0-$sub-level sets of convex functions is also convex. Therefore, this problem can be posed as the following convex optimization problem:

$$\begin{array}{ll} \min\limits_{x=(t,u)\in\mathcal{X}} & t \\ & \mathcal{X} = \{(t,u) | f(u) \leq t, g_1(u) \leq 0, g_2(u) \leq 0, \ldots, g_m(u) \leq 0\} \end{array} \tag{3.68}$$

The set $\mathcal{X}$ is convex, and hence the problem in (3.68) is a convex optimization problem. Further, every locally optimal point is also globally optimal. The computation time of algorithms for solving convex optimization problems is roughly proportional to $max\left(n^2, n^2m, C\right)$, where C is the cost of evaluating f, the g_i's and their first and second derivatives. There are many reliable and efficient algorithms for solving convex optimization problems. However, it is often difficult to recognize convex optimization problems in practice.

Examples

Consider the optimization problem

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = x_1^2 + x_2^2 \\ \text{subject to} & g_1(\mathbf{x}) = \frac{x_1}{1+x_2^2} \leq 0 \\ & h(\mathbf{x}) = (x_1 + x_2)^2 = 0 \end{array} \tag{3.69}$$

We note that the optimiation problem above is not a convex problem according to our definition, since g_1 is not convex and h is not affine. However, we note that the feasible set $\{(x_1, x_2) \mid x_1 = -x_2,\ x_1 \leq 0\}$ is convex (recall that the converse of theorem 58 does not hold - the 0-sublevel set of a non convex function can be convex). This problem can be posed as an equivalent (but not identical) convex optimization problem:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = x_1^2 + x_2^2 \\ \text{subject to} & x_1 \leq 0 \\ & x_1 + x_2 = 0 \end{array} \tag{3.70}$$

3.4 Duality Theory

Duality is a very important component of nonlinear and linear optimization models. It has a wide spectrum of applications that are very popular. It arises in the basic form of linear programming as well as in interior point methods for linear programming. The duality in linear programming was first observed by Von Neumann, and later formalized by Tucker, Gale and Kuhn. In the first attempt at extending duality beyond linear programs, duals of quadratic programs were next developed. It was subsequently observed that you can always write a dual for any optimization problem and the modern Lagrange-based 'constructive'[19] duality theory followed in the late 1960s.

An extremely popular application of duality happens to be in the quadratic programming for Support Vector Machines. The primal and dual both happen to be convex optimization programs in this case. The Minimax theorem[20], a fundamental theorem of Game Theory, proved by John von Neumann in 1928, is but one instance of the general duality theory. In the consideration of equilibrium in electrical networks, current are 'primal variables' and the potential differences are the 'dual variables'. In models of economic markets, the 'primal' variables are production and consumption levels while the 'dual' variables are prices (of goods, *etc.*). Dual price-based decomposition methods were developed by Danzig. In the case of thrust structures in mechanics, forces are primal variables and the displacements are the dual variables. Dual problems and their solutions are used for proving optimality of solutions, finding near-optimal solutions, analysing how sensitive the solution of the primal is to perturbations in the right hand side of constraints, analysing convergence of algorithms, *etc.*

3.4.1 Lagrange Multipliers

Consider the following quadratic function of $\mathbf{x} \in \Re^n$.

$$F(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} - \mathbf{x}^T \mathbf{b} \tag{3.71}$$

where A is an $n \times n$ square matrix. Consider the unconstrained minimization problem

[19] As we will see, the theory helps us construct duals that are useful in practice.
[20] The name Minimax was invented by Tucker.

$$\min_{\mathbf{x}\in\mathcal{D}} F(\mathbf{x}) \tag{3.72}$$

A locally optimum solution $\hat{\mathbf{x}}$ to this objective can be obtained by setting $\nabla F(\hat{\mathbf{x}}) = \mathbf{0}$. This condition translates to $A\hat{\mathbf{x}} = \mathbf{b}$. A sufficient condition for $\hat{\mathbf{x}}$ to be a point of local minimum is that $\nabla^2 F(\hat{\mathbf{x}}) \succ 0$. This condition holds *iff*, $A \succ 0$, that is, A is a positive definite matrix. Given that $A \succ 0$, A must be invertible (*c.f.* Section 2.12.2) and the unique solution is $\mathbf{x} = A^{-1}\mathbf{b}$.

Now suppose we have a constrained minimization problem

$$\begin{array}{ll} \min\limits_{\mathbf{y}\in\Re^n} & \frac{1}{2}\mathbf{y}^T B\mathbf{y} \\ \text{subject to} & A^T\mathbf{y} = \mathbf{b} \end{array} \tag{3.73}$$

where $\mathbf{y} \in \Re^n$, A is an $n \times m$ matrix, B is an $n \times n$ matrix and $\mathbf{b}$ is a vector of size m. To handle constrained minimization, let us consider minimization of the modified objective function $L(\mathbf{y}, \lambda) = \frac{1}{2}\mathbf{y}^T B\mathbf{y} + \lambda^T(A^T\mathbf{y} - \mathbf{b})$.

$$\min_{\mathbf{y}\in\Re^n, \lambda\in\Re^m} \quad \frac{1}{2}\mathbf{y}^T B\mathbf{y} + \lambda^T(A^T\mathbf{y} - \mathbf{b}) \tag{3.74}$$

The function $L(\mathbf{y}, \lambda)$ is called the lagrangian and involves the lagrange multiplier $\lambda \in \Re^m$. A sufficient condition for optimality of $L(\mathbf{y}, \lambda)$ at a point $L(\mathbf{y}^*, \lambda^*)$ is that $\nabla L(\mathbf{y}^*, \lambda^*) = 0$ and $\nabla^2 L(\mathbf{y}^*, \lambda^*) \succ 0$. For this particular problem:

$$\nabla L(\mathbf{y}^*, \lambda^*) = \begin{bmatrix} B\mathbf{y}^* + A\lambda^* \\ A^T\mathbf{y}^* - \mathbf{b} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and

$$\nabla^2 L(\mathbf{y}^*, \lambda^*) = \begin{bmatrix} B & A \\ A^T & 0 \end{bmatrix} \succ 0$$

The point $(\mathbf{y}^*, \lambda^*)$ must therefore satisfy, $A^T\mathbf{y}^* = \mathbf{b}$ and $A\lambda^* = -B\mathbf{y}^*$. If B is taken to be the identity matrix, $n = 2$ and $m = 1$, the minimization problem (3.73) amounts to finding a point $\mathbf{y}^*$ on a line $a_{11}y_1 + a_{12}y_2 = b$ that is closest to the origin. From geometry, we know that the point on a line closest to the origin is the point of intersection $\mathbf{p}^*$ of a perpendicular from the origin to the line. On the other hand, the solution for the minimum of (3.74), for these conditions coincides with $\mathbf{p}^*$ and is given by:

$$y_1 = \frac{a_{11}b}{(a_{11})^2 + (a_{12})^2}$$
$$y_2 = \frac{a_{12}b}{(a_{11})^2 + (a_{12})^2}$$

That is, for $n = 2$ and $m = 1$, the solution to (3.74) is the same as the solution to (3.72) Can this construction be used to always find optimal solutions to a minimization problem? We will answer this question by first motivating the concept of lagrange multipliers and in Section 3.4.2, we will formalize the lagrangian dual.

Lagrange Multipliers with Equality Constraints

The concept of lagrange multipliers can be attributed to the mathematician Lagrange, who was born in the year 1736 in Turin. He largely worked on mechanics, the calculus of variations probability, group theory, and number theory. He was party to the choice of base 10 for the metric system (rather than 12). We will here give a brief introduction to lagrange multipliers; Section 3.4.2 will discuss the *Karush-Kuhn-Tucker conditions*, which are a generalization of lagrange multipliers.

Consider the equality constrained minimization problem (with $\mathcal{D} \subseteq \Re^n$)

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in\mathcal{D}} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) = 0 & i = 1, 2, \ldots, m \end{array} \tag{3.75}$$

A direct approach to solving this problem is to find a parametrization of the constraints (as in the example on page 174) such that f is expressed in terms of the parameters, to give an unconstrained problem. For example if there is a single constraint of the form $\mathbf{x}^T A\mathbf{x} = k$, and $A \succ 0$, then the coordinate system can be rotated and $\mathbf{x}$ can be rescaled so that we get the constraint $\mathbf{y}'\mathbf{y} = k$. Further, we can substitute with parametrization of the y_i's as

$$\begin{aligned} y_1 &= k \sin\theta_1 \sin\theta_2 \ldots \sin\theta_{n-1} \\ y_2 &= k \sin\theta_1 \sin\theta_2 \ldots \cos\theta_{n-1} \\ &\ldots\ldots\ldots \end{aligned}$$

However, this is not possible for general constraints. The method of lagrange multipliers presents an indirect approach to solving this problem.

Consider a schematic representation of the problem in (3.75) with a single constraint, *i.e.*, $m = 1$ in Figure 3.39. The figure shows some level curves of the function f. The constraint function g_1 is also plotted with dotted lines in the same figure. The gradient of the constraint ∇g_1 is not parallel to the gradient ∇f of the function[21] at $f = 10.4$; it is therefore possible to move along the constraint surface so as to further reduce f. However, as shown in Figure 3.39, ∇g_1 and ∇f are parallel at $f = 10.3$, and any motion along $g_1(\mathbf{x}) = 0$ will

[21] Note that the (negative) gradient at a point is orthogonal to the contour line going through that point. This was proved in Theorem 44.

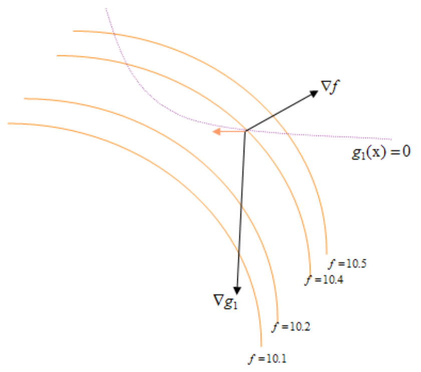


Figure 3.39: At any non-optimal and non-saddle point of the equality constrained problem, the gradient of the constraint will not be parallel to that of the function.

increase f, or leave it unchanged. Hence, at the solution $\mathbf{x}^*$, $\nabla f(\mathbf{x}^*)$ must be proportional to $-\nabla g_1(\mathbf{x}^*)$, yielding, $\nabla f(\mathbf{x}^*) = -\lambda \nabla g_1(\mathbf{x}^*)$, for some constant $\lambda \in \Re$; λ is called a *Lagrange multiplier*. In several problems, the value of λ itself need never be computed and therefore λ is often qualified as the *undetermined* lagrange multiplier.

The necessary condition for an optimum at $\mathbf{x}^*$ for the optimization problem in (3.75) with $m = 1$ can be stated as in (3.76), where the gradient is now $n+1$ dimensional with its last component being a partial derivative with respect to λ.

$$\nabla L(\mathbf{x}^*, \lambda^*) = \nabla f(\mathbf{x}^*) + \lambda^* \nabla g_1(\mathbf{x}^*) = 0 \tag{3.76}$$

The solutions to (3.76) are the stationary points of the lagrangian L; they are not necessarily local extrema of L. L is unbounded: given a point $\mathbf{x}$ that doesn't lie on the constraint, letting $\lambda \rightarrow \pm\infty$ makes L arbitrarily large or small. However, under certain stronger assumptions, as we shall see in Section 3.4.2, if the *strong Lagrangian principle* holds, the minima of f minimize the Lagrangian globally.

We will extend the necessary condition for optimality of a minimization problem with single constraint to minimization problems with multiple equality constraints (*i.e.*, $m > 1$. in (3.75)). Let $\mathcal{S}$ be the subspace spanned by $\nabla g_i(\mathbf{x})$ at any point $\mathbf{x}$ and let $\mathcal{S}_\perp$ be its orthogonal complement. Let $(\nabla f)_\perp$ be the component of ∇f in the subspace $\mathcal{S}_\perp$. At any solution $\mathbf{x}^*$, it must be true that the gradient of f has $(\nabla f)_\perp = 0$ (*i.e.*, no components that are perpendicular to all of the ∇g_i), because otherwise you could move $\mathbf{x}^*$ a little in that direction (or in the opposite direction) to increase (decrease) f without changing any of the g_i, *i.e.* without violating any constraints. Hence for multiple equality constraints, it must be true that at the solution $\mathbf{x}^*$, the space $\mathcal{S}$ contains the

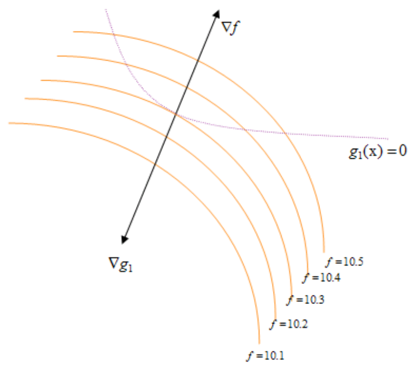


Figure 3.40: At the equality constrained optimum, the gradient of the constraint must be parallel to that of the function.

vector ∇f, *i.e.*, there are some constants λ_i such that $\nabla f(\mathbf{x}^*) = \lambda_i \nabla g_i(\mathbf{x}^*)$. We also need to impose that the solution is on the correct constraint surface (*i.e.*, $g_i = 0$, $\forall i$). In the same manner as in the case of $m = 1$, this can be encapsulated by introducing the Lagrangian $L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \nabla g_i(\mathbf{x})$, whose gradient with respect to both $\mathbf{x}$, and λ vanishes at the solution.

This gives us the following necessary condition for optimality of (3.75):

$$\nabla L(\mathbf{x}^*, \lambda^*) = \nabla \left(f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right) = 0 \tag{3.77}$$

Lagrange Multipliers with Inequality Constraints

Instead of a single equality constraint $g_1(\mathbf{x}) = 0$, we could have a single inequality constraint $g_1(\mathbf{x}) \leq 0$. The entire region labeled $g_1(\mathbf{x}) \leq 0$ in Figure 3.41 then becomes feasible. At the solution $\mathbf{x}^*$, if $g_1(\mathbf{x}^*) = 0$, *i.e.*, if the constraint is active, we must have (as in the case of a single equality constraint) that ∇f is parallel to ∇g_1, by the same argument as before. Additionally, it is necessary that the two gradients must point in opposite directions; otherwise a move away from the surface $g_1 = 0$ and into the feasible region would further reduce f. Since we are minimizing f, if the Lagrangian is written as $L = f + \lambda g_1$, we must have $\lambda \geq 0$. Therefore, with an inequality constraint, the sign of λ is important, and $\lambda \geq 0$ becomes a constraint.

However, if the constraint is not active at the solution $\nabla f(\mathbf{x}^*) = 0$, then removing g_1 makes no difference and we can drop it from $L = f + \lambda g_1$, which is equivalent to setting $\lambda = 0$. Thus, whether or not the constraints $g_1 = 0$ are active, we can find the solution by requiring that the gradients of the Lagrangian vanish, and also requiring that $\lambda g_1(\mathbf{x}^*) = 0$. This latter condition is one of the

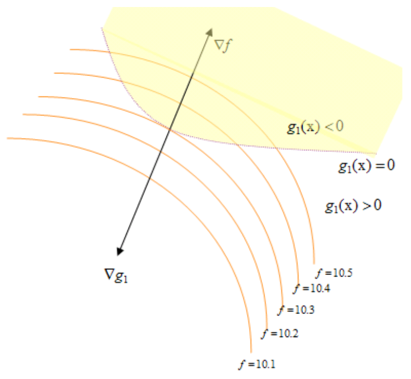


Figure 3.41: At the inequality constrained optimum, the gradient of the constraint must be parallel to that of the function.

important Karush-Kuhn-Tucker conditions of convex optimization theory that can facilitate the search for the solution and will be more formally discussed in Section 3.4.2.

Now consider the general inequality constrained minimization problem

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in\mathcal{D}} & f(\mathbf{x}) & \\ \text{subject to} & g_i(\mathbf{x}) \leq 0 & i = 1, 2, \ldots, m \end{array} \tag{3.78}$$

With multiple inequality constraints, for constraints that are active, as in the case of multiple equality constraints, ∇f must lie in the space spanned by the ∇g_i's, and if the Lagrangian is $L = f + \sum_{i=1}^{m} \lambda_i g_i$, then we must additionally have $\lambda_i \geq 0$, $\forall i$ (since otherwise f could be reduced by moving into the feasible region). As for an inactive constraint g_j ($g_j < 0$), removing g_j from L makes no difference and we can drop ∇g_j from $\nabla f = -\sum_{i=1}^{m} \lambda_i \nabla g_i$ or equivalently set $\lambda_j = 0$. Thus, the above KKT condition generalizes to $\lambda_i g_i(\mathbf{x}^*) = 0$, $\forall i$. The necessary condition for optimality of (3.78) is summarily given as

$$\begin{aligned} \nabla L(\mathbf{x}^*, \lambda^*) = \nabla \left(f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \right) &= 0 \\ \forall i \quad \lambda_i g_i(\mathbf{x}) &= 0 \end{aligned} \tag{3.79}$$

A simple and often useful trick called the *free constraint gambit* is to solve ignoring one or more of the constraints, and then check that the solution satisfies those constraints, in which case you have solved the problem.

3.4.2 The Dual Theory for Constrained Optimization

Consider the general inequality constrained minimization problem in (3.78), restated below.

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\mathcal{D}} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq 0,\ i = 1, 2, \ldots, m \end{array} \tag{3.80}$$

There are three simple and straightforward steps in forming a dual problem.

1. The first step involves forming the lagrange function by associating a price λ_i, called a lagrange multiplier, with the constraint involving g_i.

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i g_i(\mathbf{x}) = f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

2. The second step is the construction of the dual function $L^*(\lambda)$ which is defined as:

$$L^*(\lambda) = \min_{\mathbf{x}\in\mathcal{D}} L(\mathbf{x}, \lambda) = \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) + \lambda^T \mathbf{g}(\mathbf{x})$$

   What makes the theory of duality constructive is when we can solve for L^* efficiently - either in a closed form or some other 'simple' mechanism. If L^* is not easy to evaluate, the duality theory will be less useful.

3. We finally define the dual problem:

$$\begin{array}{ll} \max\limits_{\lambda\in\Re^m} & L^*(\lambda) \\ \text{subject to} & \lambda \geq \mathbf{0} \end{array} \tag{3.81}$$

It can be immediatly proved that the dual problem is a concave maximization problem.

Theorem 65 *The dual function $L^*(\lambda)$ is concave.*

Proof: Consider two values of the dual variables, *viz.*, $\lambda_1 \geq \mathbf{0}$ and $\lambda_2 \geq \mathbf{0}$. Let $\lambda = \theta\lambda_1 + (1-\theta)\lambda_2$ for any $\theta \in [0, 1]$. Then,

$$\begin{aligned} L^*(\lambda) &= \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) + \lambda^T g(\mathbf{x}) \\ &= \min_{\mathbf{x}\in\mathcal{D}} \theta\left[f(\mathbf{x}) + \lambda_1^T g(\mathbf{x})\right] + (1-\theta)\left[f(\mathbf{x}) + \lambda_2^T g(\mathbf{x})\right] \\ &\geq \min_{\mathbf{x}\in\mathcal{D}} \theta\left[f(\mathbf{x}) + \lambda_1^T g(\mathbf{x})\right] + \min_{\mathbf{x}\in\mathcal{D}} (1-\theta)\left[f(\mathbf{x}) + \lambda_2^T g(\mathbf{x})\right] \\ &= \theta L^*(\lambda_1) + (1-\theta) L^*(\lambda_2) \end{aligned}$$

This proves that $L^*(\lambda)$ is a concave function. □

The dual is concave (or the negative of the dual is convex) irrespective of the primal. Solving the dual is therefore always a convex programming problem. Thus, in some sense, the dual is better structured than the primal. However, the dual cannot be drastically simpler than the primal. For example, if the primal is not an LP, the dual cannot be an LP. Similarly, the dual can be quadratic only if the primal is quadratic.

A tricky thing in duality theory is to decide what we call the domain or *ground set* $\mathcal{D}$ and what we call the constraints g_i's. Based on whether constraints are explicitly stated or implicitly stated in the form of the ground set, the dual problem could be very different. Thus, many duals are possible for the given primal.

We will look at two examples to give a flavour of how the duality theory works.

1. We will first look at linear programming.

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & -A\mathbf{x}+\mathbf{b}\leq\mathbf{0} \end{array}$$

The lagrangian for this problem is:

$$L(\mathbf{x},\lambda)=\mathbf{c}^T\mathbf{x}+\lambda^T\mathbf{b}-\lambda^TA\mathbf{x}=\mathbf{b}^T\lambda+\left(\mathbf{c}^T-A^T\lambda\right)\mathbf{x}$$

The next step is to get L^*, which we obtain using the first derivative test:

$$L^*(\lambda)=\min_{\mathbf{x}\in\Re^n}\mathbf{b}^T\lambda+\left(\mathbf{c}^T-A\lambda\right)^T\mathbf{x}=\begin{cases}\mathbf{b}^T\lambda & \text{if } A^T\lambda=\mathbf{c}\\ -\infty & \text{if } A^T\lambda\neq\mathbf{c}\end{cases}$$

The function L^* can be thought of as the extended value extension of the same function restricted to the domain $\left\{\lambda|A^T\lambda=\mathbf{c}\right\}$. Therefore, the dual problem can be formulated as:

$$\begin{array}{ll} \max\limits_{\lambda\in\Re^m} & \mathbf{b}^T\lambda \\ \text{subject to} & A^T\lambda=\mathbf{c} \\ & \lambda\geq\mathbf{0} \end{array} \tag{3.82}$$

This is the dual of the standard LP. What if the original LP was the following?

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & -A\mathbf{x}+\mathbf{b}\leq\mathbf{0} \quad \mathbf{x}\geq\mathbf{0} \end{array}$$

Now we have a variety of options based on what constraints are introduced into the ground set (or domain) and what are explicitly treated as constraints. Some working out will convince us that treating $\mathbf{x}\in\Re^n$ as the constraint and the explicit constraints as part of the ground set is a very bad idea. One dual for this problem is the same as (3.82).

2. Let us look at a modified version of the problem in (3.83).

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & \mathbf{c}^T\mathbf{x}-\sum_{i=1}^n \ln x_i \\ \text{subject to} & -A\mathbf{x}+\mathbf{b}=\mathbf{0} \\ & \mathbf{x}>\mathbf{0} \end{array}$$

Typically, when we try to formulate a dual problem, we look for constraints that get in the way of conveniently solving the problem. We first formulate the lagrangian for this problem.

$$L(\mathbf{x},\lambda)=\mathbf{c}^T\mathbf{x}-\sum_{i=1}^n \ln x_i+\lambda^T\mathbf{b}-\lambda^TA\mathbf{x}=\mathbf{b}^T\lambda+\mathbf{x}^T\left(\mathbf{c}-A^T\lambda\right)-\sum_{i=1}^n \ln x_i$$

The domain (or ground set) for this problem is $\mathbf{x}>\mathbf{0}$, which is open.

The expression for L^* can be obtained using the first derivative test, while keeping in mind that L can be made arbitrarily small (tending to $-\infty$) unless $(\mathbf{c}-A^T\lambda)>\mathbf{0}$. This is because, even if one component of $\mathbf{c}-A^T\lambda$ is less than or equal to zero, the value of L can be made arbitrarily small by decreasing the value of the corresponding component of $\mathbf{x}$ in the $\sum_{i=1}^n \ln x_i$ part. Further, the sum $\mathbf{b}^T\lambda+\left(\mathbf{c}-A^T\lambda\right)^T\mathbf{x}-\sum_{i=1}^n \ln x_i$ can be separated out into the individual components of λ_i, and this can be exploited while determining the critical point of L.

$$L^*(\lambda)=\min_{\mathbf{x}>\mathbf{0}} \mathbf{b}^T\lambda+n+\sum_{i=1}^n \ln\frac{1}{\left(\mathbf{c}-A^T\lambda\right)_i}=\begin{cases} \mathbf{b}^T\lambda & \text{if } (\mathbf{c}-A^T\lambda)>\mathbf{0} \\ -\infty & \text{otherwise} \end{cases}$$

Finally, the dual will be

$$\begin{array}{ll} \max\limits_{\lambda\in\Re^m} & \mathbf{b}^T\lambda+n+\sum_{i=1}^n \ln\frac{1}{\left(\mathbf{c}-A^T\lambda\right)_i} \\ \text{subject to} & -A^T\lambda+\mathbf{c}>\mathbf{0} \end{array}$$

As noted earlier, the theory of duality remains a theory unless the dual lends itself to some constructive evaluation; not always is the dual a useful form.

The following *Weak duality theorem* states an important relationship between solutions to the primal (3.80) and the dual (3.81) problems.

Theorem 66 *If $p^* \in \Re$ is the solution to the primal problem in (3.80) and $d^* \in \Re$ is the solution to the dual problem in (3.81), then*

$$p^* \geq d^*$$

In general, if $\widehat{\mathbf{x}}$ is any feasible solution to the primal problem (3.80) and $\widehat{\lambda}$ is a feasible solution to the dual problem (3.81), then

$$f(\widehat{\mathbf{x}}) \geq L^*(\widehat{\lambda})$$

Proof: If $\widehat{\mathbf{x}}$ is a feasible solution to the primal problem (3.80) and $\widehat{\lambda}$ is a feasible solution to the dual problem, then

$$f(\widehat{\mathbf{x}}) \geq f(\widehat{\mathbf{x}}) + \widehat{\lambda}^T\mathbf{g}(\widehat{\lambda}) \geq \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x} + \widehat{\lambda}^T\mathbf{g}(\lambda) = L^*(\widehat{\lambda})$$

This proves the second part of the theorem. A direct consequence of this is that

$$p^* = \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) \geq \min_{\lambda\geq\mathbf{0}} L^*(\lambda) = d^*$$

□

The weak duality theorem has some important implications. If the primal problem is unbounded below, that is, $p^* = -\infty$, we must have $d^* = -\infty$, which means that the Lagrange dual problem is infeasible. Conversely, if the dual problem is unbounded above, that is, $d^* = \infty$, we must have $p^* = \infty$, which is equivalent to saying that the primal problem is infeasible. The difference, $p^* - d^*$ is called the duality gap.

In many hard combinatorial optimization problems with duality gaps, we get good dual solutions, which tell us that we are guaranteed of being some k % within the optimal solution to the primal, for some satisfactorily low values of k. This is one of the powerful uses of duality theory; constructing bounds for optimization problems.

Under what conditions can one assert that $d^* = p^*$? The condition $d^* = p^*$ is called *strong duality* and it does not hold in general. It usually holds for convex problems but there are exceptions to that - one of the most typical being that of the semi-definite optimization problem. The semi-definite program (SDP) is defined, with the linear matrix inequality constraint (*c.f.* page 206) as follows:

$$\begin{aligned}
\min_{\mathbf{x}\in\Re^n} \quad & \mathbf{c}^T\mathbf{x} \\
\text{subject to} \quad & x_1A_1 + \ldots + x_nA_n + G \preceq 0 \\
& A\mathbf{x} = \mathbf{b}
\end{aligned} \tag{3.83}$$

Sufficient conditions for strong duality in convex problems are called *constraint qualifications.* One of the most useful sufficient conditions for strong duality is called the *Slaters constraint qualification.*

Definition 52 [Slaters constraint qualification]: *For a convex problem*

$$\begin{array}{lll} \min\limits_{\mathbf{x}\in\mathcal{D}} & f(\mathbf{x}) & \\ \textit{subject to} & g_i(\mathbf{x}) \leq 0, \quad i = 1,\ldots,m & \\ & A\mathbf{x} = \mathbf{b} & \\ \textit{variable } \mathbf{x} = (x_1,\ldots,x_n) & & \end{array} \tag{3.84}$$

strong duality holds (that is $d^* = p^*$) if it is strictly feasible. That is,

$$\exists \mathbf{x} \in int(\mathcal{D}) : \quad g_i(\mathbf{x}) < 0 \quad i = 1,2,\ldots,m \quad A\mathbf{x} = \mathbf{b}$$

However, if any of the g_i's are linear, they do not need to hold with strict inequalities.

Table 3.4 summarizes some optimization problems, their duals and conditions for strong duality. Strong duality also holds for nonconvex problems

Problem type	**Objective Function**	**Constraints**	$L^*(\lambda)$	**Dual constraints**	**Strong duality**
Linear Program	$\mathbf{c}^T\mathbf{x}$	$A\mathbf{x} \leq \mathbf{b}$	$-\mathbf{b}^T\lambda$	$A^T\lambda + \mathbf{c} = \mathbf{0}$ $\lambda \geq \mathbf{0}$	Feasible primal and dual
Quadratic Program	$\frac{1}{2}\mathbf{x}^TQ\mathbf{x} + \mathbf{c}^T\mathbf{x}$ for $Q \in \mathcal{S}^n_{++}$	$A\mathbf{x} \leq \mathbf{b}$	$-\frac{1}{2}\left(\mathbf{c} - A^T\lambda\right)^T Q^{-1}\left(\mathbf{c} - A^T\lambda\right) + \mathbf{b}^T\lambda$	$\lambda \geq \mathbf{0}$	Always
Entropy maximization	$x_i\sum_{i=1}^n \ln x_i$	$A\mathbf{x} \leq \mathbf{b}$ $\mathbf{x}^T\mathbf{1} = 1$	$-\mathbf{b}^T\lambda - \mu - e^{-\mu-1}\sum_{i=1}^n e^{-\mathbf{a}_i^T\lambda}$ $\mathbf{a}_i$ is the i^{th} column of A	$\lambda \geq \mathbf{0}$	Primal constraints are satisfied.

Table 3.4: Examples of functions and their duals.

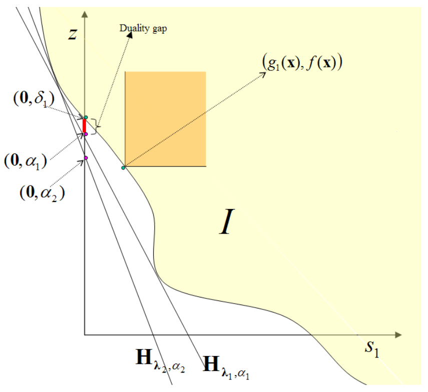
in extremely rare cases. One example of this is minimization of a nonconvex quadratic function over the unit ball.

3.4.3 Geometry of the Dual

We will study the geometry of the dual in the *column space* $\Re^{m+1}$. The column geometry of the dual will require definition of the following set:

$$\mathcal{I} = \{(\mathbf{s}, z) \mid \mathbf{s} \in \Re^m,\ z \in \Re,\ \exists \mathbf{x} \in \mathcal{D} \textit{ with } g_i(\mathbf{x}) \leq s_i\ \forall 1 \leq i \leq m,\ f(\mathbf{x}) \leq z\}$$

The set $\mathcal{I}$ is a subset of $\Re^{m+1}$, where m is the number of constraints. Consider a plot in two dimensions, for $n = 1$, with s_1 along the $x-$axis and z along the $y-$axis. For every point, $\mathbf{x} \in \mathcal{D}$, we can identify all points (s_1, z) for $s_1 \geq g_1(\mathbf{x})$



z
Duality gap
$(g_1(\mathbf{x}), f(\mathbf{x}))$
$(\mathbf{0}, \delta_1)$
$(\mathbf{0}, \alpha_1)$
$(\mathbf{0}, \alpha_2)$
I
s_1
$\mathbf{H}_{\lambda_2,\alpha_2}$
$\mathbf{H}_{\lambda_1,\alpha_1}$

Figure 3.42: Example of the set $\mathcal{I}$ for a single constraint (*i.e.*, for $n = 1$).

and $z \geq f(\mathbf{x})$ and these are points that lie to the left and above the point $(g_1(\mathbf{x}), f(\mathbf{x}))$. An example set $\mathcal{I}$ is shown in Figure 3.42. It turns out that all the intuitions we need are in two dimensions, which makes it fairly convenient to understand the idea. It is straightforward to prove that if the objective function $f(\mathbf{x})$ is convex and each of the constraints $g_i(\mathbf{x}),\ 1 \leq i \leq n$ is a convex function, then $\mathcal{I}$ must be a convex set. Since the feasible region for the primal problem (3.78) is the region in $\mathcal{I}$ with $\mathbf{s} \leq \mathbf{0}$, and since all points above and to the right of a point in $\mathcal{I}$ also belong to $\mathcal{I}$, the solution to the primal problem corresponds to the point in $\mathcal{I}$ with $\mathbf{s} = \mathbf{0}$ and least possible value of z. For example, in Figure 3.42, the solution to the primal corresponds to $(\mathbf{0}, \delta_1)$.

Let us define a hyerplane $\mathcal{H}_{\lambda,\alpha}$, parametrized by $\lambda \in \Re^m$ and $\alpha \in \Re$ as

$$\mathcal{H}_{\lambda,\alpha} = \left\{(\mathbf{s}, z) \,\middle|\, \lambda^T.\mathbf{s} + z = \alpha\right\}$$

Consider all hyperplanes that lie below $\mathcal{I}$. For example, in the Figure 3.42, both hyperplanes $\mathcal{H}_{\lambda_1,\alpha_1}$ and $\mathcal{H}_{\lambda_2,\alpha_2}$ lie below the set $\mathcal{I}$. Of all hyperplanes that lie below $\mathcal{I}$, consider the hyperplane whose intersection with the line $\mathbf{s} = \mathbf{0}$, corresponds to as high a value of z as possible. This hyperplane must be supporting hyperplane. Incidentally, $\mathcal{H}_{\lambda_1,\alpha_1}$ happens to be such a supporting hyperplane. Its point of intersection $(\mathbf{0}, \alpha_1)$ precisely corresponds to the solution to the dual problem. Let us derive this statement formally after setting up some more notation.

We will define two half-spaces corresponding to $\mathcal{H}_{\lambda,\alpha}$

$$\mathcal{H}^+_{\lambda,\alpha} = \left\{(\mathbf{s}, z) \,\middle|\, \lambda^T.\mathbf{s} + z \geq \alpha\right\}$$

$$\mathcal{H}^-_{\lambda,\alpha} = \left\{(\mathbf{s}, z) \,\middle|\, \lambda^T.\mathbf{s} + z \leq \alpha\right\}$$

Let us define another set $\mathcal{L}$ as

$$\mathcal{L} = \{(\mathbf{s}, z) \,|\mathbf{s} = \mathbf{0}\}$$

Note that $\mathcal{L}$ is essentially the z or function axis. The intersection of $\mathcal{H}_{\lambda,\alpha}$ with $\mathcal{L}$ is the point $(\mathbf{0}, \alpha)$. That is

$$(\mathbf{0}, \alpha) = \mathcal{L} \bigcap \mathcal{H}_{\lambda,\alpha}$$

We would like to manipulate λ and α so that the set $\mathcal{I}$ lies in the half-space $\mathcal{H}^{+}_{\lambda,\alpha}$ as tightly as possible. Mathematically, we are interested in the problem of maximizing the height of the point of intersection of $\mathcal{L}$ with $\mathcal{H}_{\lambda,\alpha}$ above the $\mathbf{s} = \mathbf{0}$ plane, while ensuring that $\mathcal{I}$ remains a subset of $\mathcal{H}^{+}_{\lambda,\alpha}$.

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & \mathcal{H}^{+}_{\lambda,\alpha} \supseteq \mathcal{I} \end{array}$$

By definitions of $\mathcal{I}$, $\mathcal{H}^{+}_{\lambda,\alpha}$ and the subset relation, this problem is equivalent to

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & \lambda^T.\mathbf{s} + z \geq \alpha \ \forall (\mathbf{s}, z) \in \mathcal{I} \end{array}$$

Now notice that if $(\mathbf{s}, z) \in \mathcal{I}$, then $(\mathbf{s}', z) \in \mathcal{I}$ for all $\mathbf{s}' \geq \mathbf{s}$. This was also illustrated in Figure 3.42. Thus, we cannot afford to have any component of λ negative; if any of the λ_i's were negative, we could cranck up s_i arbitrarily to violate the inequality $\lambda^T.\mathbf{s} + z \geq \alpha$. Thus, we can add the constraint $\lambda \geq \mathbf{0}$ to the above problem without changing the solution.

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & \lambda^T.\mathbf{s} + z \geq \alpha \ \forall (\mathbf{s}, z) \in \mathcal{I} \\ & \lambda \geq \mathbf{0} \end{array}$$

Any equality constraint $h(\mathbf{x}) = 0$ can be expressed using two inequality constraints, *viz.*, $h(\mathbf{x}) \leq 0$ and $-h(\mathbf{x}) \leq 0$, implying that its corresponding lagrange multiplier should be exactly 0. This problem can again be proved to be equivalent to the following problem, using the fact that every point on $\partial\mathcal{I}$ must be of the form $(g_1(\mathbf{x}), g_2(\mathbf{x}), \ldots, g_m(\mathbf{x}), f(\mathbf{x}))$ for some $\mathbf{x} \in \mathcal{D}$.

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & \lambda^T.\mathbf{g}(\mathbf{x}) + f(\mathbf{x}) \geq \alpha \ \forall \mathbf{x} \in \mathcal{D} \\ & \lambda \geq \mathbf{0} \end{array}$$

We will remind the reader at this point that $L(\mathbf{x}, \lambda) = \lambda^T.\mathbf{g}(\mathbf{x}) + f(\mathbf{x})$. The above problem is therefore the same as

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & L(\mathbf{x}, \lambda) \geq \alpha \ \forall \mathbf{x} \in \mathcal{D} \\ & \lambda \geq \mathbf{0} \end{array}$$

Since, $L^*(\lambda) = \min_{\mathbf{x} \in \mathcal{D}} L(\mathbf{x}, \lambda)$, we can deal with the equivalent problem

$$\begin{array}{ll} \max & \alpha \\ \text{subject to} & L^*(\lambda) \geq \alpha \\ & \lambda \geq \mathbf{0} \end{array}$$

This problem can be restated as

$$\begin{array}{ll} \max & L^*(\lambda) \\ \text{subject to} & \lambda \geq \mathbf{0} \end{array}$$

This is precisely the dual problem. We thus get a geometric interpretation of the dual.

Again referring to Figure 3.42, we note that if the set $\mathcal{I}$ is not convex, there could be a *gap* between the $z-$intercept $(\mathbf{0}, \alpha_1)$ of the best supporting hyperplane $\mathcal{H}_{\lambda_1,\alpha_1}$ and the closest point $(\mathbf{0}, \delta_1)$ of $\mathcal{I}$ on the $z-$axis, which corresponds to the solution to the primal. In fact, when the set $\mathcal{I}$ is not convex, we can never prove that there will be no duality gap. And even when the set $\mathcal{I}$ is convex, bizzaire things can happen; for example, in the case of semi-definite programming, the set $\mathcal{I}$, though convex, is not at all well-behaved and this yields a large duality gap, as shown in Figure 3.43. In fact, the set $\mathcal{I}$ is open from below (the dotted boundary) for a semi-definite program. We could create very simple problems with convex $\mathcal{I}$, for which there are duality gaps. For well-behaved convex functions (as in the case of linear programming), there are no duality gaps. Figure 3.44 illustrates the case of a well-behaved convex program.

3.4.4 Complementary slackness and KKT Conditions

We now state the conditions between the primal and dual optimal points for an arbitrary function. These conditions, called the *Karush-Kuhn-Tucker conditions* (abbreviated as KKT conditions) state a necessary condition for a solution to be optimal with zero duality gap. Consider the following general optimization problem.

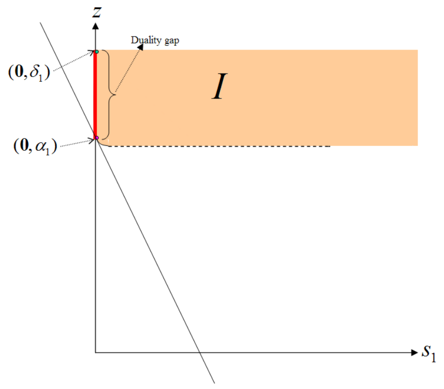


Figure 3.43: Example of the convex set $\mathcal{I}$ for a single constrained semi-definite program.

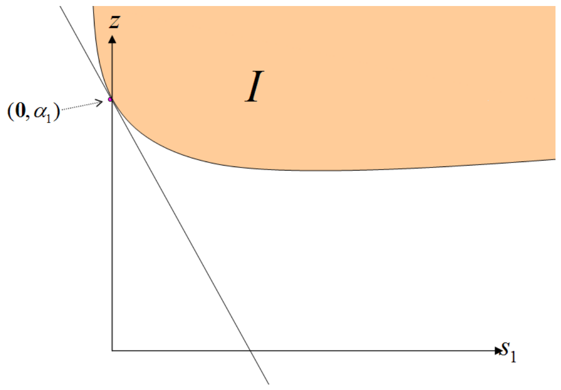


Figure 3.44: Example of the convex set $\mathcal{I}$ for a single constrained well-behaved convex program.

$$
\begin{array}{lll}
\min\limits_{\mathbf{x}\in\mathcal{D}} & f(\mathbf{x}) & \\
\text{subject to} & g_i(\mathbf{x}) \leq 0, & i = 1,\ldots,m \\
 & h_j(\mathbf{x}) = 0, & j = 1,\ldots,p \\
\text{variable } \mathbf{x} = (x_1,\ldots,x_n) & &
\end{array}
\tag{3.85}
$$

Suppose that the primal and dual optimal values for the above problem are attained and equal, that is, strong duality holds. Let $\widehat{\mathbf{x}}$ be a primal optimal and $(\widehat{\lambda}, \widehat{\mu})$ be a dual optimal point ($\widehat{\lambda} \in \Re^m, \widehat{\mu} \in \Re^p$). Thus,

$$
\begin{aligned}
f(\widehat{\mathbf{x}}) &= L^*(\widehat{\lambda}, \widehat{\mu}) \\
&= \min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}) + \widehat{\lambda}^T \mathbf{g}(\mathbf{x}) + \widehat{\mu}^T \mathbf{h}(\mathbf{x}) \\
&\leq f(\widehat{\mathbf{x}}) + \widehat{\lambda}^T \mathbf{g}(\widehat{\mathbf{x}}) + \widehat{\mu}^T \mathbf{h}(\widehat{\mathbf{x}}) \\
&\leq f(\widehat{\mathbf{x}})
\end{aligned}
$$

The last inequality follows from the fact that $\widehat{\lambda} \geq \mathbf{0}$, $\mathbf{g}(\widehat{\mathbf{x}}) \leq \mathbf{0}$, and $\mathbf{h}(\widehat{\mathbf{x}}) = \mathbf{0}$. We can therefore conclude that the two inequalities in this chain must hold with equality. Some of the conclusions that we can draw from this chain of equalities are

1. That $\widehat{\mathbf{x}}$ is a minimizer for $L(\mathbf{x}, \widehat{\lambda}, \widehat{\mu})$ over $\mathbf{x} \in \mathcal{D}$. In particular, if the functions f, $g_1, g_2, \ldots, g_m$ and $h_1, h_2, \ldots, h_p$ are differentiable (and therefore have open domains), the gradient of $L(\mathbf{x}, \widehat{\lambda}, \widehat{\mu})$ must vanish at $\widehat{\mathbf{x}}$, since any point of global optimum must be a point of local optimum. That is,

$$
\nabla f(\widehat{\mathbf{x}}) + \sum_{i=1}^{m} \widehat{\lambda}_i \nabla g_i(\widehat{\mathbf{x}}) + \sum_{j=1}^{p} \widehat{\mu}_j \nabla h_j(\widehat{\mathbf{x}}) = \mathbf{0} \tag{3.86}
$$

2. That

$$
\widehat{\lambda}^T \mathbf{g}(\widehat{\mathbf{x}}) = \sum_{i=1}^{n} \widehat{\lambda}_i g_i(\widehat{\mathbf{x}}) = 0
$$

Since each term in this sum is nonpositive, we conclude that

$$
\widehat{\lambda}_i g_i(\widehat{\mathbf{x}}) = 0 \; for \; i = 1, 2, \ldots, m \tag{3.87}
$$

This condition is called *complementary slackness* and is a necessary condition for strong duality. Complementary slackness implies that the i^{th}

optimal lagrange multiplier is 0 unless the i^{th} inequality constraint is active at the optimum. That is,

$$\begin{array}{lcl} \widehat{\lambda}_i > 0 & \Rightarrow & g_i(\widehat{\mathbf{x}}) = 0 \\ g_i(\widehat{\mathbf{x}}) < 0 & \Rightarrow & \widehat{\lambda}_i = 0 \end{array}$$

Let us further assume that the functions f, $g_1, g_2, \ldots, g_m$ and $h_1, h_2, \ldots, h_p$ are differentiable on open domains. As above, let $\widehat{\mathbf{x}}$ be a primal optimal and $(\widehat{\lambda}, \widehat{\mu})$ be a dual optimal point with zero duality gap. Putting together the conditions in (3.86), (3.87) along with the feasibility conditions for any primal solution and dual solution, we can state the following Karush-Kuhn-Tucker (KKT) necessary conditions for zero duality gap.

$$\begin{array}{lrcll} (1) & \nabla f(\widehat{\mathbf{x}}) + \sum_{i=1}^{m} \widehat{\lambda}_i \nabla g_i(\widehat{\mathbf{x}}) + \sum_{j=1}^{p} \widehat{\mu}_j \nabla h_j(\widehat{\mathbf{x}}) & = & \mathbf{0} & \\ (2) & g_i(\widehat{\mathbf{x}}) & \leq & 0 & i = 1, 2, \ldots, m \\ (3) & \widehat{\lambda}_i & \geq & 0 & i = 1, 2, \ldots, m \\ (4) & \widehat{\lambda}_i g_i(\widehat{\mathbf{x}}) & = & 0 & i = 1, 2, \ldots, m \\ (5) & h_j(\widehat{\mathbf{x}}) & = & 0 & j = 1, 2, \ldots, p \end{array} \tag{3.88}$$

When the primal problem is convex, the KKT conditions are also sufficient for the points to be primal and dual optimal with zero duality gap. If f is convex, g_i are convex and h_j are affine, the primal problem is convex and consequently, the KKT conditions are sufficient conditions for zero duality gap.

Theorem 67 *If the function f is convex, g_i are convex and h_j are affine, then KKT conditions in 3.88 are necessary and sufficient conditions for zero duality gap.*

Proof: The necessity part has already been proved; here we only prove the sufficiency part. The conditions (2) and (5) in (3.88) ensure that $\widehat{\mathbf{x}}$ is primal feasible. Since $\lambda \geq \mathbf{0}$, $L(\mathbf{x}, \widehat{\lambda}, \widehat{\mu})$ is convex in $\mathbf{x}$. Based on condition (1) in (3.88) and theorem 62, we can infer that $\widehat{\mathbf{x}}$ minimizes $L(\mathbf{x}, \widehat{\lambda}, \widehat{\mu})$. We can thus conclude that

$$\begin{aligned} L^*(\widehat{\lambda}, \widehat{\mu}) &= f(\widehat{\mathbf{x}}) + \widehat{\lambda}^T \mathbf{g}(\widehat{\mathbf{x}}) + \widehat{\mu}^T \mathbf{h}(\widehat{\mathbf{x}}) \\ &= f(\widehat{\mathbf{x}}) \end{aligned}$$

In the equality above, we use $h_j(\widehat{\mathbf{x}}) = 0$ and $\widehat{\lambda}_i g_i(\widehat{\mathbf{x}}) = 0$. Further,

$$d^* \geq L^*(\widehat{\lambda}, \widehat{\mu}) == f(\widehat{\mathbf{x}}) \geq p^*$$

The duality theorem (theorem 66) however states that $p^* \geq d^*$. This implies that

$$d^* = L^*(\widehat{\lambda}, \widehat{\mu}) == f(\widehat{\mathbf{x}}) = p^*$$

This shows that $\widehat{\mathbf{x}}$ and $(\widehat{\lambda}, \widehat{\mu})$ correspond to the primal and dual optimals respectively and the problem therefore has zero duality gap. $\Box$

In summary, for any convex optimization problem with differentiable objective and constraint functions, any points that satisfy the KKT conditions are primal and dual optimal, and have zero duality gap.

The KKT conditions play a very important role in optimization. In some rare cases, it is possible to solve the optimization problems by finding a solution to the KKT conditions analytically. Many algorithms for convex optimization are conceived as, or can be interpreted as, methods for solving the KKT conditions.

3.5 Algorithms for Unconstrained Minimization

We will now study some algorithms for solving convex problems. These techniques are relevant for most convex optimization problems that do not yield themselves to closed form solutions. We will start with unconstrained minimization.

Recall that the goal in unconstrained minimization is to solve the convex problem

$$\min_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x})$$

We are not interested in f, whose solution can be obtained in closed form. For example, minimizing a quadratic is very simple and can be solved by linear equations, an example of which was discussed in Section 2.9.2. Let us denote the optimal solution of the minimization problem by p^*. We will assume that f is convex and twice continuously differentiable and that it attains a finite optimal value p^*. Most unconstrained minimization techniques produce a sequence of points $\mathbf{x}^{(k)} \in \mathcal{D}, k = 0, 1, \ldots$ such that $f\left(\mathbf{x}^{(k)}\right) \to p^*$ as $k \to \infty$ or, $\nabla f\left(\mathbf{x}^{(k)}\right) \to \mathbf{0}$ as $k \to \infty$. Iterative techniques for optimization, further require a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$ and sometimes that $epi(f)$ is closed. The $epi(f)$ can be inferred to be closed either if $\mathcal{D} = \Re^n$ or $f(\mathbf{x}) \to \infty$ as $\mathbf{x} \to \partial\mathcal{D}$. The function $f(x) = \frac{1}{x}$ for $x > 0$ is an example of a function whose $epi(f)$ is closed.

While there exist convergence proofs (including guarantees on number of optimization iterations) for many convex optimization algorithms, the proofs assume many conditions,many of which are either not verifiable or involve unknown constants (such as the Lipshitz constant). Thus, most convergence proofs for convex optimization problems are useless in practice, though it is good to know that there are conditions under which the algorithm converges. Since convergence proofs are only of theoretical importance, we will make the strongest possible assumption under which convergence can be proved easily, which is that the function f is strongly convex (*c.f.* Section 3.2.7 for definition of strong convexity) with the strong convexity constant $c > 0$ for which $\nabla^2 f(\mathbf{x}) \succeq cI \; \forall x \in \mathcal{D}$.

Further, it can be proved that for a strongly convex function f, $\nabla^2 f(\mathbf{x}) \preceq DI$ for some constant $D \in \Re$. The ratio $\frac{D}{c}$ is an upper bound on the condition number of the matrix $\nabla^2 f(\mathbf{x})$.

3.5.1 Descent Methods

Descent methods for unconstrained optimization have been in use since the last 70 years or more. The general idea in descent methods is that the next iterate $\mathbf{x}^{(k+1)}$ is the current iterate $\mathbf{x}^{(k)}$ added with a descent or search direction $\Delta\mathbf{x}^{(k)}$ (a unit vector), which is multiplied by a scale factor $t^{(k)}$, called the step length.

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$$

The incremental step is determined while ensuring that $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$. We assume that we are dealing with the extended value extension of the convex function f (*c.f.* definition 46), which returns ∞ for any point outside its domain. However, if we do so, we need to make sure that the initial point indeed lies in the domain $\mathcal{D}$.

A single iteration of the general descent algorithm (shown in Figure 3.45) consists of two main steps, *viz.*, determining a good descent direction $\Delta\mathbf{x}^{(k)}$, which is typically forced to have unit norm and determining the step size using some line search technique. If the function f is convex, and we require that $f(\mathbf{x}^{(k+1)}) < f(\mathbf{x}^{(k)})$ then, we must have $\nabla^T f(\mathbf{x}^{(k+1)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) < 0$. This can be seen from the necessary and sufficient condition for convexity stated in equation (3.44) within Section 3.2.9 and restated here for reference.

$$f(\mathbf{x}^{(k+1)}) \geq f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)})$$

Since $t^{(k)} > 0$, we must have

$$\nabla^T f(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} < 0$$

That is, the descent direction $\Delta\mathbf{x}^{(k)}$ must make an obtuse angle ($\theta \in \left(\frac{\pi}{2}, \frac{3\pi}{2}\right)$) with the gradient vector.

```
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$
repeat
    1. Determine $\Delta\mathbf{x}^{(k)}$.
    2. Choose a step size $t^{(k)} > 0$ using ray[a] search.
    3. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
    4. Set $k = k + 1$.
until stopping criterion (such as $||\nabla f(\mathbf{x}^{(k+1)})|| < \epsilon$) is satisfied
```

[a] Many textbooks refer to this as line search, but we prefer to call it ray search, since the step must be positive.

Figure 3.45: The general descent algorithm.

There are many different empirical techniques for ray search, though it matters much less than the search for the descent direction. These techniques reduce the $n-$dimensional problem to a $1-$dimensional problem, which can be easy to solve by use of plotting and eyeballing or even exact search.

1. **Exact ray search:** The exact ray search seeks a scaling factor t that satisfies

$$t = \underset{t>0}{\operatorname{argmin}} f(\mathbf{x} + t\Delta\mathbf{x}) \tag{3.89}$$

2. **Backtracking ray search:** The exact line search may not be feasible or could be expensive to compute for complex non-linear functions. A relatively simpler ray search iterates over values of step size starting from 1 and scaling it down by a factor of $\beta \in (0, \frac{1}{2})$ after every iteration till the following condition, called the *Armijo condition* is satisfied for some $0 < c_1 < 1$.

$$f(\mathbf{x} + t\Delta\mathbf{x}) < f(\mathbf{x}) + c_1 t \nabla^T f(\mathbf{x}) \Delta\mathbf{x} \tag{3.90}$$

Based on equation (3.44), it can be inferred that the Armijo inequality can never hold for $c_1 = 1$; for $c_1 = 1$, the right hand side of the Armijo condition gives a lower bound on the value of $f(\mathbf{x} + t\Delta\mathbf{x})$. The Armijo condition simply ensures that t decreases f sufficiently. Often, another condition is used for inexact line search in conjunction with the Armijo condition.

$$\left|\Delta\mathbf{x}^T \nabla f(\mathbf{x} + t\Delta\mathbf{x})\right| \leq c_2 \left|\Delta\mathbf{x}^T \nabla f(\mathbf{x})\right| \tag{3.91}$$

where $1 > c_1 > c_2 > 0$. This condition ensures that the slope of the function $f(\mathbf{x}+t\Delta\mathbf{x})$ at t is less than c_2 times that at $t = 0$. The conditions in (3.90) and (3.91) are together called the strong Wolfe conditions. These conditions are particularly very important for non-convex problems.

A finding that is borne out of plenty of empirical evidence is that exact ray search does better than empirical ray search in a few cases only. Further, the exact choice of the value of β and α seems to have little effect on the convergence of the overall descent method.

The trend of specific descent methods has been like a parabola - starting with simple steepest descent techniques, then accomodating the curvature hessian matrix through a more sophisticated Newton's method and finally, trying to simplify the Newton's method through approximations to the hessian inverse, culminating in conjugate gradient techniques, that do away with any curvature matrix whatsoever, and form the internal combustion engine of many sophisticated optimization techniques today. We start the thread by describing the steepest descent methods.

Steepest Descent

Let $\mathbf{v} \in \Re^n$ be a unit vector under some norm. By theorem 60, for convex f,

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \mathbf{v}) \leq -\nabla^T f(\mathbf{x}^{(k)})\mathbf{v}$$

For small $\mathbf{v}$, the inequality turns into approximate equality. The term $-\nabla^T f(\mathbf{x}^{(k)})\mathbf{v}$ can be thought of as (an upper-bound on) the first order prediction of decrease. The idea in the steepest descent method [**?**] is to choose a norm and then determine a descent direction such that for a unit step in that norm, the first order prediction of decrease is maximized. This choice of the descent direction can be stated as

$$\Delta\mathbf{x} = \text{argmin}\left\{\nabla^T f(\mathbf{x})\mathbf{v} \mid ||\mathbf{v}|| = 1\right\}$$

The algorithm is outlined in Figure 3.46.

```
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$.
repeat
  1. Set $\Delta\mathbf{x}^{(k)} = \text{argmin}\left\{\nabla^T f(\mathbf{x}^{(k)})\mathbf{v} \mid ||\mathbf{v}|| = 1\right\}$.
  2. Choose a step size $t^{(k)} > 0$ using exact or backtracking ray search.
  3. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
  4. Set $k = k + 1$.
until stopping criterion (such as $||\nabla f(\mathbf{x}^{(k+1)})|| \leq \epsilon$) is satisfied
```

Figure 3.46: The steepest descent algorithm.

The key to understanding the steepest descent method (and in fact many other iterative methods) is that it heavily depends on the choice of the norm. It has been empirically observed that if the norm chosen is aligned with the gross geometry of the sub-level sets[22], the steepest descent method converges faster to the optimal solution. If the norm chosen is not aligned, it often amplifies the effect of oscillations. Two examples of the steepest descent method are the gradient descent method (for the eucledian or L_2 norm) and the coordinate-descent method (for the L_1 norm). One fact however is that no two norms should give exactly opposite steepest descent directions, though they may point in different directions.

Gradient Descent

A classic greedy algorithm for minimization is the gradient descent algorithm. This algorithm uses the negative of the gradient of the function at the current

[22]The alignment can be determined by fitting, for instance, a quadratic to a sample of the points.

point $\mathbf{x}^*$ as the descent direction $\Delta\mathbf{x}^*$. It turns out that this choice of $\Delta\mathbf{x}^*$ corresponds to the direction of steepest descent under the L_2 (eucledian) norm. This can be proved in a straightforward manner using theorem 43. The algorithm is outlined in Figure 3.47. The steepest descent method can be thought

```
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$
repeat
  1. Set $\Delta\mathbf{x}^{(k)} = -\nabla f(\mathbf{x}^{(k)})$.
  2. Choose a step size $t^{(k)} > 0$ using exact or backtracking ray search.
  3. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
  4. Set $k = k + 1$.
until stopping criterion (such as $||\nabla f(\mathbf{x}^{(k+1)})||_2 \leq \epsilon$) is satisfied
```

Figure 3.47: The gradient descent algorithm.

of as changing the coordinate system in a particular way and then applying the gradient descent method in the changed coordinate system.

Coordinate-Descent Method

The co-ordinate descent method corresponds exactly to the choice of L_1 norm for the steepest descent method. The steepest descent direction using the L_1 norm is given by

$$\Delta\mathbf{x} = -\frac{\partial f(\mathbf{x})}{\partial x_i}\mathbf{u}^i$$

where,

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = ||\nabla f(\mathbf{x})||_\infty$$

and $\mathbf{u}^i$ was defined on page 175 as the unit vector pointing along the i^{th} coordinate axis. Thus each iteration of the coordinate descent method involves optimizing over one component of the vector $\mathbf{x}^{(k)}$ and then updating the vector. The component chosen is the one having the largest absolute value in the gradient vector. The algorithm is outlined in Figure 3.48.

Convergence of Steepest Descent Method

For the gradient method, it can be proved that if f is strongly convex,

$$f(\mathbf{x}^{(k)}) - p^* \leq \rho^k \left(f(\mathbf{x}^{(0)} - p^* \right) \tag{3.92}$$

The value of $\rho \in (0, 1)$ depends on the strong convexity constant c (*c.f.* equation (3.64) on page 221), the value of $\mathbf{x}^{(0)}$ and type of ray search employed. The suboptimality $f(\mathbf{x}^{(k)}) - p^*$ goes down by a factor $\rho < 1$ at every step and this is referred to as *linear convergence*[23]. However, this is only of theoretical

[23] A series $s_1, s_2, \ldots$ is said to have

```
**Find** a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$.
**Select** an appropriate norm $||.||$.
**repeat**
  1. Let $\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_i^{(k)}} = ||\nabla f(\mathbf{x}||_\infty$ .
  2. Set $\Delta\mathbf{x}^{(k)} = -\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_i^{(k)}}\mathbf{u}^i$.
  3. Choose a step size $t^{(k)} > 0$ using exact or backtracking ray search.
  4. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
  5. Set $k = k + 1$.
**until** stopping criterion (such as $||\nabla f(\mathbf{x}^{(k+1)})||_\infty \leq \epsilon$) is satisfied
```

Figure 3.48: The coordinate descent algorithm.

importance, since this method is often very slow, indicated by values of ρ, very close to 1. Use of exact line search in conjunction with gradient descent also has the tendency to overshoot the next best iterate. It is therefore rarely used in practice. The convergence rate depends greatly on the condition number of the Hessian (which is upperbounded by $\frac{D}{c}$). It can be proved that the number of iterations required for the convergence of the gradient descent method is lower-bounded by the condition number of the hessian; large eigenvalues correspond to high curvature directions and small eigenvalues correspond to low curvature directions. Many methods (such as conjugate gradient) try to improve upon the gradient method by making the hessian better conditioned. Convergence can be very slow even for moderately well-conditioned problems, with condition number in the 100s, even though computation of the gradient at each step is only an $O(n)$ operation. The gradient descent method however works very well if the function is isotropic, that is if the level-curves are spherical or nearly spherical.

The convergence of the steepest descent method can be stated in the same form as in 3.92, using the fact that any norm can be bounded in terms of the Euclidean norm, *i.e.*, there exists a constant $\eta \in (0,1]$ such that

$$||\mathbf{x}|| \geq \eta||\mathbf{x}||_2$$

1. linear convergence to $\overline{s}$ if $\lim\limits_{i\to\infty} \frac{|s_{i+1}-\overline{s}|}{|s_i-\overline{s}|} = \delta \in (0,1)$. For example, $s_i = (\gamma)^i$ has linear convergence to $\overline{s} = 0$ for any $\gamma < 1$. The rate of decrease is also sometimes called exponential or geometric. This is considered quite slow.

2. superlinear convergence to $\overline{s}$ if $\lim\limits_{i\to\infty} \frac{|s_{i+1}-\overline{s}|}{|s_i-\overline{s}|} = 0$. For example, $s_i = \frac{1}{i!}$ has superlinear convergence. This is the most common.

3. quadratic convergence to $\overline{s}$ if $\lim\limits_{i\to\infty} \frac{|s_{i+1}-\overline{s}|}{|s_i-\overline{s}|^2} = \delta \in (0,\infty)$. For example, $s_i = (\gamma)^{2^i}$ has quadratic convergence to $\overline{s} = 0$ for any $\gamma < 1$. This is considered very fast in practice.

3.5.2 Newton's Method

Newton's method [?] is based on approximating a function around the current iterate $\mathbf{x}^{(k)}$ using a second degree Taylor expansion.

$$f(\mathbf{x}) \approx \widetilde{f}(\mathbf{x}) = f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}-\mathbf{x}^{(k)})$$

If the function f is convex, the quadratic approximation is also convex. Newton's method is based on solving it exactly by finding its critical point $\mathbf{x}^{(k+1)}$ as a function of $\mathbf{x}^{(k)}$. Setting the gradient of this quadratic approximation (with respect to $\mathbf{x}$) to $\mathbf{0}$ gives

$$\nabla^T f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = 0$$

solving which yields the next iterate as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)}) \tag{3.93}$$

assuming that the Hessian matrix is invertible. The term $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ can be thought of as an update step. This leads to a simple descent algorithm, outlined in Figure 3.49 and is called the Newton's method. It relies on the invertibility of the hessian, which holds if the hessian is positive definite as in the case of a strictly convex function. In case the hessian is invertible,cholesky factorization (page 150) of the hessian can be used to solve the linear system (3.93). However, the Newton method may not even be properly defined if the hessian is not positive definite. In this case, the hessian could be changed to a nearby positive definite matrix whenever it is not. Or a line search could be added to seek a new point having a positive definite hessian.

This method uses a step size of 1. If instead, the stepsize is chosen using exact or backtracking ray search, the method is called the *damped Newton's method*. Each Newton's step takes $O(n^3)$ time (without using any fast matrix multiplication methods).

The Newton step can also be looked upon as another incarnation of the steepest descent rule, but with the quadratic norm defined by the (local) Hessian $\nabla^2 f(\mathbf{x}^{(k)})$ evaluated at the current iterate $\mathbf{x}^{(k)}$, *i.e.*,

$$||\mathbf{u}||_{\nabla^2 f(\mathbf{x}^{(k)})} = \left(\mathbf{u}\nabla^2 f(\mathbf{x}^{(k)})\mathbf{u}\right)^{\frac{1}{2}}$$

The norm of the Newton step, in the quadratic norm defined by the Hessian at a point $\mathbf{x}$ is called the *Newton decrement* at the point $\mathbf{x}$ and is denoted by $\lambda(\mathbf{x})$. Thus,

$$\lambda(\mathbf{x}) = ||\Delta\mathbf{x}||_{\nabla^2 f(\mathbf{x})} = \nabla^T f(\mathbf{x}) \left(\nabla^2 f(\mathbf{x})\right)^{-1} \nabla f(\mathbf{x})$$

The Newton decrement gives an 'estimate' of the proximity of the current iterate $\mathbf{x}$ to the optimal point $\mathbf{x}^*$ obtained by measuring the proximity of $\mathbf{x}$ to the

```
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$.
Select an appropriate tolerance $\epsilon > 0$.
repeat
  1. Set $\Delta\mathbf{x}^{(k)} = -\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x})$.
  2. Let $\lambda^2 = \nabla^T f(\mathbf{x}^{(k)}) \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)})$.
  3. If $\frac{\lambda^2}{2} \leq \epsilon$, quit.
  4. Set step size $t^{(k)} = 1$.
  5. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
  6. Set $k = k + 1$.
until
```

Figure 3.49: The Newton's method.

minimum point of the quadratic approximation $\widetilde{f}(\mathbf{x})$. The estimate is $\frac{1}{2}\lambda(\mathbf{x})^2$ and is given as

$$\frac{1}{2}\lambda(\mathbf{x})^2 = f(\mathbf{x}) - \min\ \widetilde{f}(\mathbf{x})$$

Additionally, $\lambda(\mathbf{x})^2$ is also the directional derivative in the Newton direction.

$$\lambda(\mathbf{x})^2 = \nabla^T f(\mathbf{x})\Delta\mathbf{x}$$

The estimate $\frac{1}{2}\lambda(\mathbf{x})^2$ is used to test the convergence of the Newton algorithm in Figure 3.49.

Next, we state an important property of the Newton's update rule.

Theorem 68 *If $\Delta\mathbf{x}^{(k)} = -\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)})$, $\nabla^2 f(\mathbf{x}^{(k)})$ is symmetric and positive definite and $\Delta\mathbf{x}^{(k)} \neq \mathbf{0}$, then $\Delta\mathbf{x}^{(k)}$ is a descent direction at $\mathbf{x}^{(k)}$, that is, $\nabla^T f(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} < 0$.*

Proof: First of all, if $\nabla^2 f(\mathbf{x}^{-1})$ is symmetric and positive definite, then it is invertible and its inverse is also symmetric and positive definite. Next, we see that

$$\nabla^T f(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = -\nabla^T f(\mathbf{x}) \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)}) < 0$$

because $\left(\nabla^2 f(\mathbf{x})\right)^{-1}$ is symmetric and positive definite. □

The Newton method is independent of affine changes of coordinates. That is, if optimizating a function $f\mathbf{x})$ using the Newton's method with an initial estimate $\mathbf{x}^{(0)}$ involves the series of iterates $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(k)}, \ldots$, then optimizing the same problem using the Newton's method with a change of coordinates given by $\mathbf{x} = A\mathbf{y}$ and the intial estimate $\mathbf{y}^{(0)}$ such that $\mathbf{x}^{(0)} = A\mathbf{y}^{(0)}$ yields the series of iterates $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots, \mathbf{y}^{(k)}, \ldots$, such that $\mathbf{x}^{(k)} = A\mathbf{y}^{(k)}$. This is a great advantage over the gradient method, whose convergence can be very sensitive to affine transformation.

Another well known feature of the Newton's method is that it converges very fast, if at all. The convergence is extremely fast in the vicinity of the point of optimum. This can be loosely understood as follows. If $\mathbf{x}^*$ is the critical point of a differentiable convex function f, defined on an open domain, the function is approximately equal to its second order taylor approximation in the vicinity of $\mathbf{x}^*$. Further, $\nabla f(\mathbf{x}^*) = \mathbf{0}$. This gives the following approximation at any point $\mathbf{x}$ in the vicinity of $\mathbf{x}^*$.

$$\begin{aligned} f(\mathbf{x}) &\approx f(\mathbf{x}^*) + \nabla^T f(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T \nabla^2 f(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) \\ &= f(\mathbf{x}^*) + \frac{1}{2}(\mathbf{x}-\mathbf{x}^*)^T \nabla^2 f(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) \end{aligned}$$

Thus, the level curves of a convex function are approximately ellipsoids near the point of minimum $\mathbf{x}^*$. Given this geometry near the minimum, it then makes sense to do steepest descent in the norm induced by the hessian, near the point of minimum (which is equivalent to doing a steepest descent after a rotation of the coordinate system using the hessian). This is exactly the Newton's step. Thus, the Newton's method[24] converges very fast in the vicinity of the solution.

This convergence analysis is formally stated in the following theorem and is due to Leonid Kantorovich.

Theorem 69 *Suppose $f(\mathbf{x}) : \mathcal{D} \rightarrow \Re$ is twice continuously differentiable on $\mathcal{D}$ and $\mathbf{x}^*$ is the point corresponding to the optimal value p^* (so that $\nabla f(\mathbf{x}^*) = 0$). Let f be strongly convex on $\mathcal{D}$ with constant $c > 0$. Also, suppose $\nabla^2 f(\mathbf{x}^*)$ is Lipschitz continuous on $\mathcal{D}$ with a constant $L > 0$ (which measures how well f can be approximated by a quadratic function or how fast the second derivative of f changes), that is*

$$||\nabla^2 f(\mathbf{x}) - \nabla^2 f(\mathbf{y})||_2 \leq L||\mathbf{x}-\mathbf{y}||_2$$

Then, there exist constants $\alpha \in (0, \frac{c^2}{L})$ and $\beta > 0$ such that

1. **Damped Newton Phase:** *If $||\nabla^2 f(\mathbf{x})||_2 \geq \alpha$, then $f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)}) \leq -\beta$. That is, at every step of the iteration in the damped Newton phase, the function value decreases by atleast β and the phase ends after at most $\frac{f(\mathbf{x}^{(0)})-p^*}{\beta}$ iterations, which is a finite number.*

2. **Quadratically Convergent Phase:** *If $||\nabla^2 f(\mathbf{x})||_2 < \alpha$, then $\frac{L}{2c^2}||\nabla f(\mathbf{x}^{(k+1)})||_2 \leq \left(\frac{L}{2c^2}||\nabla f(\mathbf{x}^{(k)})||_2\right)^2$. When applied recursively this inequality yields*

$$\frac{L}{2c^2}||\nabla f(\mathbf{x}^{(k)})||_2 \leq \left(\frac{1}{2}\right)^{2^{k-q}}$$

where q is iteration number, starting at which $||\nabla^2 f(\mathbf{x}^{(q)})||_2 < \alpha$. Using the result for strong convexity in equation (3.50) on page 217, we can derive

[24] Newton originally presented his method for one-dimensional problems. Later on Raphson extended the method to multi-dimensional problems.

$$f(\mathbf{x}^{(k)}) - p^* \leq \frac{1}{2c}||\nabla f(\mathbf{x}^{(k)})||_2^2 \leq \frac{2c^3}{L^2}\left(\frac{1}{2}\right)^{2^{k-q+1}} \tag{3.94}$$

Also, using the result in equation (3.52) on page 217, we get a bound on the distance between the current iterate and the point $\mathbf{x}^*$ *corresponding to the optimum.*

$$||\mathbf{x}^{(k)} - \widehat{\mathbf{x}}^*||_2 \leq \frac{2}{c}||\nabla f(\mathbf{x}^{(k)})||_2 \leq \frac{c}{L}\left(\frac{1}{2}\right)^{2^{k-q}} \tag{3.95}$$

Inequality (3.94) shows that convergence is quadratic once the second condition is satisfied after a finite number of iterations. Roughly speaking, this means that, after a sufficiently large number of iterations, the number of correct digits doubles at each iteration[25]. In practice, once in the quadratic phase, you do not even need to bother about any convergence criterion; it suffices to apply a fixed few number of Newton iterations to get a very accurate solution. Inequality (3.95) states that the sequence of iterates converges quadratically. The Lipschitz continuity condition states that if the second derviative of the function changes relatively slowly, applying Newton's method can be useful. Again, the inequalities are technical junk as far as practical application of Newton's method is concerned, since L, c and α are generally unknown, but it helps to understand the properties of the Newton's method, such as its two phases and identify them in problems. In practice, Newton's method converges very rapidly, if at all.

As an example, consider a one dimensional function $f(x) = 7x - \ln x$. Then $f'(x) = 7 - \frac{1}{x}$ and $f''(x) = \frac{1}{x^2}$. The Newton update rule at a point x is $x^{new} = x - x^2\left(7 - \frac{1}{x}\right)$. Starting with $x^{(0)} = 0$ is really infeasible and useless, since the updates will always be 0. The unique global minimizer of this function is $x^* = \frac{1}{7}$. The range of quadratic convergence for Newton's method on this function is $x \in \left(0, \frac{2}{7}\right)$. However, if you start with an initial infeasible point $x^{(0)} = 0$, the function will quadratically tend to $-\infty$!

There are some classes of functions for which theorem 69 can be applied very constructively. They are

- $-\sum_{i=1}^{m} \ln x_i$
- $-\ln t^2 - \mathbf{x}^T\mathbf{x}$ for $t > 0$
- $-\ln det(X)$

Further, theorem 69 also comes handy for linear combinations of these functions. These three functions are also at the heart of modern interior points method theory.

[25]Linear convergence adds a constant number of digits of accuracy at each iteration.

3.5.3 Variants of Newton's Method

One important aspect of the algorithm in Figure 3.49 is the step (1), which involves solving a linear system $\nabla^2 f(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = \nabla f(\mathbf{x}^{(k)})$. The system can be easy to solve if the Hessian is a 100×100 sparse matrix, but it can get hairy if it is a larger and denser matrix. Thus it can be unfair to claim that the Newton's method is faster than the gradient descent method on the grounds that it takes a fewer number of iterations to converge as compared to the gradient descent, since each iteration of the Newton's method involves inverting the hessian to solve a linear system, which can take time[26] $O(n^3)$ for dense systems. Further, the method assumes that the hessian is positive definite and therefore invertible, which might not always be so. Finally, the Newton's method might make huge-uncontrolled steps, especially when the hessian is positive semi-definite (for example, if the function is flat along the direction corresponding to a 0 or nearly 0 eigenvalue). Due to these disadvantages, most optimization packages do not use Newton's method.

There is a whole suite of methods called *Quasi-Newton methods* that use approximations of the hessian at each iteration in an attempt to either do less work per iteration or to handle singular hessian matrices. These methods fall in between gradient methods and Newton's method and were introduced in the 1960's. Work on quasi-Newton methods sprang from the belief that often, in a large linear system, most variables should not depend on most other variables (that is, the system is generally sparse).

We should however note that in some signal and image processing problems, the hessian has a nice structure, which allows one to solve the linear system $\nabla^2 f(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = \nabla f(\mathbf{x}^{(k)})$ in time much less than $O(n^3)$ (often in time comparble to that required for quasi Newton methods), without having to explicitly store the entire hessian. We next discuss some optimization techniques that use specific approximations to the hessian $\nabla^2 f(\mathbf{x})$ for specific classes of problems, by reducing the time required for computing the second derivatives.

3.5.4 Gauss Newton Approximation

The Gauss Newton method decomposes the objective function (typically for a regression problem) as a composition of two functions[27] $f = l \circ \mathbf{m}$; (i) the vector valued model or regression function $\mathbf{m} : \Re^n \to \Re^p$ and (ii) the scalar-valued loss (such as the sum squared difference between predicted outputs and target outputs) function l. For example, if m_i is $y_i - r(\mathbf{t}_i, \mathbf{x})$, for parameter vector $\mathbf{x} \in \Re^n$ and input instances $(y_i, \mathbf{t}_i)$ for $i = 1, 2, \ldots, p$, the function f can be written as

$$f(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{p}(y_i - r(\mathbf{t}_i, \mathbf{x}))^2$$

[26] $O(n^{2.7})$ to be precise.

[27] Here, n is the number of weights.

An example of the function r is the linear regression function $r(\mathbf{t}_i, \mathbf{x}) = \mathbf{x}^T\mathbf{t}_i$. Logistic regression poses an example objective function, which involves a cross-entropy loss.

$$f(\mathbf{x}) = -\sum_{i=1}^{p}\left(y_i \log\left(\sigma(\mathbf{x}^T\mathbf{t}_i)\right) + (1-y_i)\log\left(\sigma(-\mathbf{x}^T\mathbf{t}_i)\right)\right)$$

where $\sigma(k) = \frac{1}{1+e^{-k}}$ is the logistic function.

The task of the loss function is typically to make the optimization work well and this gives freedom in choosing l. Many different objective functions share a common loss function. While the sum-squared loss function is used in many regression settings, cross-entropy loss is used in many classification problems. These loss functions arise from the problem of maximizing log-likelihoods in some reasonable way.

The Hessian $\nabla^2 f(\mathbf{x})$ can be expressed using a matrix version of the chain rule, as

$$\nabla^2 f(\mathbf{x}) = \underbrace{J_{\mathbf{m}}(\mathbf{x})^T \nabla^2 l(\mathbf{m}) J_{\mathbf{m}}(\mathbf{x})}_{G_f(\mathbf{x})} + \sum_{i=1}^{p} \nabla^2 m_i(\mathbf{x})(\nabla l(\mathbf{m}))_i$$

where $J_{\mathbf{m}}$ is the jacobian[28] of the vector valued function $\mathbf{m}$. It can be shown that if $\nabla^2 l(\mathbf{m}) \succeq 0$, then $G_f(\mathbf{x}) \succeq 0$. The term $G_f(\mathbf{x})$ is called the Gauss-Newton approximation of the Hessian $\nabla^2 f(\mathbf{x})$. In many situtations, $G_f(\mathbf{x})$ is the dominant part of $\nabla^2 f(\mathbf{x})$ and the approximation is therefore reasonable. For example, at the point of minimum (which will be the critical point for a convex function), $\nabla^2 f(\mathbf{x}) = G_f(\mathbf{x})$. Using the Gauss-Newton approximation to the hessian $\nabla^2 f(\mathbf{x})$, the Newton update rule can be expressed as

$$\Delta\mathbf{x} = (G_f(\mathbf{x}))^{-1}\nabla f(\mathbf{x}) = (G_f(\mathbf{x}))^{-1} J_{\mathbf{m}}^T(\mathbf{x})\nabla l(\mathbf{m})$$

where we use the fact that $(\nabla f(\mathbf{x}))_i = \sum_{k=1}^{p} \frac{\partial l}{\partial m_k}\frac{\partial m_k}{\partial x_i}$, since the gradient of a composite function is a product of the jacobians.

For the cross entropy classification loss or the sum-squared regression loss l, the hessian is known to be positive semi-definite. For example, if the loss function is the sum of squared loss, the objective function is $f = \frac{1}{2}\sum_{i=1}^{p} m_i(\mathbf{x})^2$ and $\nabla^2 l(\mathbf{m}) = I$. The Newton update rule can be expressed as

$$\Delta\mathbf{x} = (J_{\mathbf{m}}(\mathbf{x})^T J_{\mathbf{m}}(\mathbf{x}))^{-1} J_{\mathbf{m}}(\mathbf{x})^T \mathbf{m}(\mathbf{x})$$

Recall that $(J_{\mathbf{m}}(\mathbf{x})^T J_{\mathbf{m}}(\mathbf{x}))^{-1} J_{\mathbf{m}}(\mathbf{x})^T$ is the Moore-Penrose pseudoinverse $J_{\mathbf{m}}(\mathbf{x})^+$ of $J_{\mathbf{m}}(\mathbf{x})$. The Gauss-Jordan method for the sum-squared loss can be interpreted as multiplying the gradient $\nabla l(\mathbf{m})$ by the pseudo-inverse of the jacobian of $\mathbf{m}$

[28]The Jacobian is a $p \times n$ matrix of the first derivatives of a vector valued function, where p is arity of $\mathbf{m}$. The $(i,j)^{th}$ entry of the Jacobian is the derivative of the i^{th} output with respect to the j^{th} variable, that is $\frac{\partial m_i}{\partial x_j}$. For $m = 1$, the Jacobian is the gradient vector.

instead of its transpose (which is what the gradient descent method would do). Though the Gauss-Newton method has been traditionally used for non-linear least squared problems, recently it has also seen use for the cross entropy loss function. This method is a simple adoption of the Newton's method, with the advantage that second derivatives, which can be computationally expensive and challenging to compute, are not required.

3.5.5 Levenberg-Marquardt

Like the Gauss-Newton method, the Levenberg-Marquardt method has its main application in the least squares curve fitting problem (as also in the minimum cross-entropy problem). The Levenberg-Marquardt method interpolates between the Gauss-Newton algorithm and the method of gradient descent. The Levenberg-Marquardt algorithm is more robust than the Gauss Newton algorithm - it often finds a solution even if it starts very far off the final minimum. On the other hand, for well-behaved functions and reasonable starting parameters, this algorithm tends to be a bit slower than the Gauss Newton algorithm. The Levenberg-Marquardt method aims to reduce the uncontrolled step size often taken by the Newton's method and thus fix the stability issue of the Newton's method. The update rule is given by

$$\Delta\mathbf{x} = -\left(G_f(\mathbf{x}) + \lambda\,\mathrm{diag}(G_f)\right)^{-}1 J_{\mathbf{m}}^T(\mathbf{x})\nabla l(\mathbf{m})$$

where G_f is the Gauss-Newton approximation to $\nabla^2 f(\mathbf{x})$ and is assumed to be positive semi-definite. This method is one of the work-horses of modern optimization. The parameter $\lambda \geq 0$ adaptively controlled, limits steps to an elliptical model-trust region[29]. This is achieved by adding λ to the smallest eigenvalues of G_f, thus restricting all eigenvalues of the matrix to be above λ so that the elliptical region has diagonals of shorter length that inversely vary as the eigenvalues (*c.f.* page 2.11.3). While this method fixes the stability issues in Newtons method, it still requires the $O(n^3)$ time required for matrix inversion.

3.5.6 BFGS

The Broyden-Fletcher-Goldfarb-Shanno[30] (BFGS) method uses linear algebra to iteratively update an estimate $B^{(k)}$ of $\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1}$ (the inverse of the curvature matrix), while ensuring that the approximation to the hessian inverse is symmetric and positive definite. Let $\Delta\mathbf{x}^{(k)}$ be the direction vector for the k^{th} step obtained as the solution to

$$\Delta\mathbf{x}^{(k)} = -B^{(k)}\nabla f(\mathbf{x}^{(k)})$$

The next point $\mathbf{x}^{(k+1)}$ is obtained as

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$$

[29]Essentially the algorithm approximates only a certain region (the so-called trust region) of the objective function with a quadratic as opposed to the entire function.

[30]The the 4 authors wrote papers for exactly the same method at exactly at the same time.

where $t^{(k)}$ is the step size obtained by line search. Let $\Delta\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$. Then the BFGS update rule is derived by imposing the following logical conditions:

1. $\Delta\mathbf{x}^{(k)} = -B^{(k)}\nabla f(\mathbf{x}^{(k)})$ with $B^{(k)} \succ 0$. That is, $\Delta\mathbf{x}^{(k)}$ is the minimizer of the convex quadratic model
$$Q^{(k)}(\mathbf{p}) = f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})\mathbf{p} + \frac{1}{2}\mathbf{p}^T\left(B^{(k)}\right)^{-1}\mathbf{p}$$

2. $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$, where $t^{(k)}$ is obtained by line search.

3. The gradient of the function $Q^{(k+1)} = f(\mathbf{x}^{(k+1)}) + \nabla^T f(\mathbf{x}^{(k+1)})\mathbf{p} + \frac{1}{2}\mathbf{p}^T\left(B^{(k+1)}\right)^{-1}\mathbf{p}$ at $\mathbf{p} = \mathbf{0}$ and $\mathbf{p} = -t^{(k)}\Delta\mathbf{x}^{(k)}$ agrees with gradient of f at $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{(k)}$ respectively. While the former condition is naturally satisfied, the latter need to be imposed. This quasi-Newton condition yields
$$\left(B^{(k+1)}\right)^{-1}\left(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\right) = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}).$$
This equation is called the *secant equation*.

4. Finally, among all symmetric matrices satisfying the secant equation, $B^{(k+1)}$ is closest to the current matrix $B^{(k)}$ in some norm. Different matrix norms give rise to different quasi-Newton methods. In particular, when the norm chosen is the Frobenius norm, we get the following BGFS update rule
$$B^{(k+1)} = B^{(k)} + R^{(k)} + S^{(k)}$$
where,
$$R^{(k)} = \frac{\Delta\mathbf{x}^{(k)}\left(\Delta\mathbf{x}^{(k)}\right)^T}{\left(\Delta\mathbf{x}^{(k)}\right)^T\Delta\mathbf{g}^{(k)}} - \frac{B^{(k)}\Delta\mathbf{g}^{(k)}\left(\Delta\mathbf{g}^{(k)}\right)^T\left(B^{(k)}\right)^T}{\left(\Delta\mathbf{g}^{(k)}\right)^T B^{(k)}\Delta\mathbf{g}^{(k)}}$$
and
$$S^{(k)} = \mathbf{u}\left(\Delta\mathbf{x}^{(k)}\right)^T B^{(k)}\Delta\mathbf{x}^{(k)}\mathbf{u}^T$$
with
$$\mathbf{u} = \frac{\Delta\mathbf{x}^{(k)}}{\left(\Delta\mathbf{x}^{(k)}\right)^T\Delta\mathbf{g}^{(k)}} - \frac{B^{(k)}\Delta\mathbf{g}^{(k)}}{\left(\Delta\mathbf{g}^{(k)}\right)^T B^{(k)}\Delta\mathbf{g}^{(k)}}$$
We have made use of the Sherman Morrison formula that determines how updates to a matrix relate to the updates to the inverse of the matrix.

The approximation to the Hessian is updated by analyzing successive gradient vectors and thus the Hessian matrix does not need to be computed at any stage. The initial estimate $B^{(0)}$ can be taken to be the identity matrix, so that the first step is equivalent to a gradient descent. The BFGS method has a reduced complexity of $O(n^2)$ time per iteration. The method is summarized

```
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$ and an approximate $B^{(0)}$ (which could be $I$).
Select an appropriate tolerance $\epsilon > 0$.
repeat
    1. Set $\Delta\mathbf{x}^{(k)} = -B^{(k)}\nabla f(\mathbf{x}^{(k)})$.
    2. Let $\lambda^2 = \nabla^T f(\mathbf{x}^{(k)})B^{(k)}\nabla f(\mathbf{x}^{(k)})$.
    3. If $\frac{\lambda^2}{2} \leq \epsilon$, quit.
    4. Set step size $t^{(k)} = 1$.
    5. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
    6. Compute $\Delta\mathbf{g}^{(k)} = \nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)})$.
    7. Compute $R^{(k)}$ and $S^{(k)}$.
    8. Compute $B^{(k+1)} = B^{(k)} + R^{(k)} + S^{(k)}$.
    6. Set $k = k + 1$.
until
```

Figure 3.50: The BFGS method.

in Figure 3.50 The BFGS [**?**] method approaches the Newton's method in behaviour as the iterate approaches the solution. They are much faster than the Newton's method in practice. It has been proved that when BFGS is applied to a convex quadratic function with exact line search, it finds the minimizer within n steps. There is a variety of methods related to BFGS and collectively they are known as Quasi-Newton methods. They are preferred over the Newton's method or the Levenberg-Marquardt when it comes to speed. There is a variant of BFGS, called LBFGS [**?**], which stands for "Limited memory BFGS method". LBFGS employs a limited-memory quasi-Newton approximation that does not require much storage or computation. It limites the rank of the inverse of the hessian to some number $\gamma \in \Re$ so that only $n\gamma$ numbers have to be stored instead of n^2 numbers. For general non-convex problems, LBFGS may fail when the initial geometry (in the form of $B^{(0)}$) has been placed very close to a saddle point. Also, LBFGS is very sensitive to line search.

Recently, L-BFGS has been observed [**?**] to be the most effective parameter estimation method for Maximum Entropy model, much better than improved iterative scaling [**?**] (IIS) and generalized iterative scaling [**?**] (GIS).

3.5.7 Solving Systems Large Sparse Systems

In many convex optimization problems such as least squares, newton's method for optimization, *etc.*, one has to deal with solving linear systems involving large and sparse matrices. Elimination with ordering can be expensive in such cases. A lot of work has gone into solving such problems efficiently[31] using iterative

[31]Packages such as LINPack (which is now renamed to LAPACK), EiSPACK, MINPACK, *etc.*, which can be found under the netlib respository, have focused on efficiently solving large linear systems under general conditions as well as specific conditions such as symmetry or positive definiteness of the coefficient matrix.

methods instead of direct elimination methods. An example iterative method is for solving a system $A\mathbf{x} = \mathbf{b}$ by repeated multiplication of a large and sparse matrix A by vectors to quickly get an answer $\widehat{\mathbf{x}}$ that is sufficiently close to the optimal solution $\mathbf{x}^*$. Multiplication of an $n \times n$ sparse matrix A having k non-zero entries with a vector of dimension n takes $O(kn)$ time only, in contrast to $O(n^3)$ time for Gauss elimination. We will study three types of methods for solving systems with large and sparse matrices:

1. *Iterative Methods.*
2. *Multigrid Methods.*
3. *Krylov Methods.*

The most famous and successful amongst the Krylov methods has been the *conjugate gradient method*, which works for problems with positive definite matrices.

Iterative Methods

The central step in an iteration is

$$P\mathbf{x}_{k+1} = (P - A)\mathbf{x}_k + \mathbf{b}$$

where $\mathbf{x}_k$ is the estimate of the solution at the k^{th} step, for $k = 0, 1, \ldots$. If the iterations converge to the solution, that is, if $\mathbf{x}_{k+1} = \mathbf{x}_k$ one can immediatly see that the solution is reached. The choice of matrix P, which is called the *preconditioner*, determines the rate of convergence of the solution sequence to the actual solution. The initial estimate $\mathbf{x}_0$ can be arbitrary for linear systems, but for non-linear systems, it is important to start with a good approximation. It is desirable to choose the matrix P reasonably close to A, though setting $P = A$ (which is referred to as perfect preconditioning) will entail solving the large system $A\mathbf{x} = \mathbf{b}$, which is undesirable as per our problem definition. If $\mathbf{x}^*$ is the actual solution, the relationship between the errors $\mathbf{e}_k$ and $\mathbf{e}_{k+1}$ at the k^{th} and $(k+1)^{th}$ steps respectively can be expressed as

$$P\mathbf{e}_{k+1} = (P - A)\mathbf{e}_k$$

where $\mathbf{e}_k = \mathbf{x}_k - \mathbf{x}^*$. This is called the *error equation.* Thus,

$$\mathbf{e}_{k+1} = (I - P^{-1}A)\mathbf{e}_k = M\mathbf{e}_k$$

Whether the solutions are convergent or not is controlled by the matrix M. The iterations are stationary (that is, the update is of the same form at every step). On the other hand, Multigrid and Krylov methods adapt themselves across iterations to enable faster convergence. The error after k steps is given by

$$\mathbf{e}_k = M^k\mathbf{e}_0 \tag{3.96}$$

Using the idea of eigenvector decomposition presented in (2.99), it can be proved that the error vector $\mathbf{e}_k \rightarrow \mathbf{0}$ if the absolute values of all the eigenvalues of M are less than 1. This is the *fundamental theorem of iteration.* In this case, the rate of convergence of $\mathbf{e}_k$ to $\mathbf{0}$ is determined by the maximum absolute eigenvalue of M, called the spectral radius of M and denoted by $\rho(M)$.

Any iterative method should attempt to choose P so that it is easy to compute $\mathbf{x}_{k+1}$ and at the same time, the matrix $M = I - P^{-1}A$ has small eigenvalues. Corresponding to various choices of the preconditioner P, there exist different iterative methods.

1. *Jacobi:* In the simplest setting, P can be chosen to be a diagonal matrix with its diagonal borrowed from A. This choice of A is correponds to the *Jacobi* method. The value of $\rho(M)$ is less than 1 for the Jacobi method, though it is often very close to 1. Thus, the Jacobi method does converge, but the convergence can be very slow in practice. While the residual $\widehat{\mathbf{r}} = A\widehat{\mathbf{x}} - \mathbf{b}$ converges rapidly, the error $\overline{\mathbf{x}} = \widehat{\mathbf{x}} - \mathbf{x}^*$ decreases rapidly in the beginning, but the rate of decrease of $\overline{\mathbf{x}}$ reduces as iterations proceed. This happens because $\overline{\mathbf{x}} = A^{-1}\widehat{\mathbf{r}}$ and A^{-1} happens to have large condition number for sparse matrices. In fact, it can be shown that Jacobi can take upto n^{β} iterations to reduce the error $\overline{\mathbf{x}}$ by a factor β.

   We will take an example to illustrate the Jacobi method. Consider the following $n \times n$ tridiagonal matrix A.

$$A = \begin{bmatrix} 2 & -1 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ldots & 0 & 0 & 0 & \ldots & 0 \\ . & . & . & \ldots & . & . & . & \ldots & . \\ . & . & . & \ldots & . & . & . & \ldots & . \\ 0 & 0 & 0 & \ldots & -1 & 2 & -1 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 & -1 & 2 & \ldots & 0 \\ . & . & . & \ldots & . & . & . & \ldots & . \\ . & . & . & \ldots & . & . & . & \ldots & . \\ 0 & 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 2 \end{bmatrix} \qquad (3.97)$$

   The absolute value of the i^{th} eigenvalue of M is $\cos \frac{j\pi}{n+1}$ and its spectral radius is $\rho(M) = \cos \frac{\pi}{n+1}$. For extremely large n, the spectral radius is approximately $1 - \frac{1}{2}\left(\frac{\pi}{n+1}\right)^2$, which is very close to 1. Thus, the Jacobi steps converge very slowly.

2. *Gauss-Seidel:* The second possibility is to choose P to be the lower-triangular part of A. The method for this choice is called the *Gauss-Siedel* method. For the example tridiagonal matrix A in (3.97), matrix

$P - A$ will be the strict but negated upper-triangular part of A. For the Gauss-Seidel technique, the components of $\mathbf{x}_{k+1}$ can be determined from $\mathbf{x}_k$ using back-substitution. The Gauss-sidel method provides only a constant factor improvement over the *Jacobi* method.

3. *Successive over-relaxation:* In this method, the preconditioner is obtained as a weighted composition of the preconditioners from the above two methods. It is abbreviated as SOR. In history, this was the first step of progress beyond Jacobi and Gauss-Seidel.

4. *Incomplete LU:* This method involves an incomplete elimination on the sparse matrix A. For a sparse matrix A, many entries in its LU decomposition will comprise of nearly 0 elements; the idea behind this method is to treat such entries as 0's. Thus, the L and U matrices are approximated based on the tolerance threshold; if the tolerance threshold is very high, the factors are exact. Else they are approximate.

Multigrid Methods

Multigrid methods come very handy in solving large sparse systems, especially differential equations using a hierarchy of discretizations. This approach often scales linearly with the number of unknowns n for a pre-specified accuracy threshold. The overall multi-grid algorithm for solving $A_h\mathbf{u}_h = \mathbf{b}_h$ with residual given by $\mathbf{r}_h = \mathbf{b} - A\mathbf{u}_h$ is

1. **Smoothing:** Perform a few (say 2-3) iterations on $A_h\mathbf{u} = \mathbf{b}_h$ using either Jacobi or Gauss-sidel. This will help remove high frequency components of the residual $\mathbf{r} = \mathbf{b} - A_h\mathbf{u}$. This step is really outside the core of the multigrid method. Denote the solution obtained by $\mathbf{u}_h$. Let $\mathbf{r}_h = \mathbf{b} - A_h\mathbf{u}_h$.

2. **Restriction:** Restrict $\mathbf{r}_h$ to coarse grid by setting $\mathbf{r}_{2h} = R\mathbf{r}_h$. That is, $\mathbf{r}_h$ is downsampled to yield $\mathbf{r}_{2h}$ Let $k < n$ characterize the coarse grid. Then, the $k \times n$ matrix R is called the restriction matrix and it takes the residuals from a finer to a coarser grid. It is typically scaled to ensure that a vector of 1's on the fine mesh gets transformed to a vector of 1's on a coarse mesh. Calculations on the coarse grid are way faster than on the finer grid.

3. Solve $A_{2h}\mathbf{e}_{2h} = \mathbf{r}_{2h}$ with $A_{2h} = RA_hN$, which is a natural construction for the coarse mesh operation. This could be done by running few iterations of Jacobi, starting with $\mathbf{e}_{2h} = \mathbf{0}$.

4. **Interpolation/Prolongation:** This step involves interpolating the correction computed on a coarses grid to a finer grid. Interpolate back to $\mathbf{e}_h = N\mathbf{e}_{2h}$. Here N is a $k \times n$ interpolation matrix and it takes the residuals from a coarse to a fine grid. It is generally a good idea to connect N to R by setting $N = \alpha R^T$ for some scaling factor α. Add $\mathbf{e}_h$ to $\mathbf{u}_h$. The

analytical expression for $\mathbf{e}_h$ is

$$\mathbf{e}_h = N(A_{2h})^{-1}RA_h(\mathbf{u}-\mathbf{u}_h) = \underbrace{\left(N(RAN)^{-1}RA_h(\mathbf{u}-\mathbf{u}_h)\right)}_{S}(\mathbf{u}-\mathbf{u}_h)$$

A property of the $n \times n$ matrix S is that $S^2 = S$. Thus, the only eigenvalues of S are 0 and 1. Since S is of rank $k < n$, k of its eigenvalues are 1 and $n-k$ are 0. Further, the eigenvectors for the 1 eigenvalues, which are in the null space of $I-S$ form the coarse mesh (and correspond to low frequency vectors) whereas the eigenvectors for the 0 eigenvalues, which are in the null space of S form the fine mesh (and correspond to high frequency vectors). We can easily derive that k eigenvalues of $I-S$ will be 0 and $n-k$ of them will be 1.

5. Finally as a post-smoothing step, iterate $A\mathbf{u}_h = \mathbf{b}_h$ starting from the improved $\mathbf{u}_h + \mathbf{e}_h$, using Jacobi or Gauss-Sidel.

Overall, the error $\mathbf{e}^k$ after k steps will be of the form

$$\mathbf{e}_k = (M^t(I-S)M^t)\mathbf{e}_0 \tag{3.98}$$

where t is the number of Jacobi steps performed in (1) and (5). Typically t is 2 or 3. When you contrast (3.98) against (3.96), we discover that $\rho(M) \geq\geq \rho(M^t(I-S)M^t)$. As t increases, $\rho(M^t(I-S)M^t)$ further decreases by a smaller proportion.

In general, you could have multiple levels of coarse grids corresponding to $2h$, $4h$, $8h$ and so on, in which case, steps (2), (3) and (4) would be repeated as many times with varying specifications of the coarseness. If A is an $n \times n$ matrix, multi-grid methods are known to run in $O(n^2)$ floating point operations (flops). The multi-grid method could be used an iterative method to solve a linear system. Alternatively, it could be used to obtain the preconditioner.

Linear Conjugate Gradient Method

The conjugate gradient method is one of the most popular Krylov methods. The Krylov matrix K_j, for the linear system $A\mathbf{u} = \mathbf{b}$ is given by

$$K_j = \begin{bmatrix}\mathbf{b} & A\mathbf{b} & A^2\mathbf{b} & \ldots & A^{j-1}\mathbf{b}\end{bmatrix}$$

The columns of K_j are easy to compute; each column is a result of a matrix multiplication A with the previous column. Assuming we are working with sparse matrices, (often symmetric matrices such as the Hessian) these computations will be inexpensive. The Krylov space $\mathcal{K}_j$ is the column space of K_j. The columns of K_j are computed during the first j steps of an iterative method such as Jacobi. Most Krylov methods opt to choose vectors from $\mathcal{K}_j$ instead of a fixed choice of the j^{th} column of K_j. A method such as *MinRes* chooses a vector

$\mathbf{u}_j \in \mathcal{K}_j$ that minimizes $\mathbf{b} - A\mathbf{u}_j$. One of the well-known Krylov methods is the *Conjugate gradient* method, which assumes that the matrix A is symmetric and positive definite and is faster than MinRes. In this method, the choice of u_j is made so that $\mathbf{b} - A\mathbf{u}_j \perp \mathcal{K}_j$. That is, the choice of $\mathbf{u}_j$ is made so that the residual $\mathbf{r}_j = \mathbf{b} - A\mathbf{u}_j$ is orthogonal to the space $\mathcal{K}_j$. The conjugate gradient method gives an exact solution to the linear system if $j = n$ and that is how they were originally designed to be (and put aside subsequently). But later, they were found to give very good approximations for $j << n$.

The discussions that follow require the computation of a basis for $\mathcal{K}_j$. It is always prefered to have a basis matrix with low condition number[32], and an orthonormal basis is a good choice, since it has a condition number of 1 (the basis consisting of the columns of K_j turns out to be not-so-good in practice). The *Arnoldi* method yields an orthonormal Krylov basis $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_j$ to get something that is numerically reasonable to work on. The method is summarized in Figure 3.51. Though the underlying idea is borrowed from Gram-Schmidt at every step, there is a difference; the vector $\mathbf{t}$ is $\mathbf{t} = A\mathbf{Q}_j$ as against simply $\mathbf{t} = \mathbf{q}_j$. Will it be expensive to compute each $\mathbf{t}$? Not if A is symmetric. First we note that by construction, $AQ = QH$, where $\mathbf{q}_j$ is the j^{th} column of Q. Thus, $H = Q^T AQ$. If A is symmetric, then so is H. Further, since H has only one lower diagonal (by construction), it must have only one higher diagonal. Therefore, H must be symmetric and tridiagonal. If A is symmetric, it suffices to subtract only the components of $\mathbf{t}$ in the direction of the last two vectors $\mathbf{q}_{j-1}$ and $\mathbf{q}_j$ from $\mathbf{t}$. Thus, for a symmetric A, the inner 'for' loop needs to iterate only over $i = j - 1$ and $i = j$.

Since A and H are similar matrices, they have exactly the same eigenvalues. Restricting the computation to a smaller number of orthonormal vectors (for some $k << n$), we can save time for computing Q_k and H_k. The k eigenvalues of H_k are good approximations to the first k eigenvalues of H. This is called the *Arnoldi-Lanczos* method for finding the top k eigenvalues of a matrix.

As an example, consider the following matrix A.

$$A = \begin{bmatrix} 0.5344 & 1.0138 & 1.0806 & 1.8325 \\ 1.0138 & 1.4224 & 0.9595 & 0.8234 \\ 1.0806 & 0.9595 & 1.0412 & 1.0240 \\ 1.8325 & 0.8234 & 1.0240 & 0.7622 \end{bmatrix}$$

[32]For any matrix A, the condition number $\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)}$, where $\sigma_{max}(A)$ and $\sigma_{min}(A)$ are maximal and minimal singular values of A respectively. Recall from Section 2.13 that the i^{th} eigenvalue of $A^T A$ (the gram matrix) is the square of the i^{th} singular value of A. Further, if A is normal, $\kappa(A) = \left|\frac{\lambda_{max}(A)}{\lambda_{min}(A)}\right|$, where $\lambda_{max}(A)$ and $\lambda_{min}(A)$ are eigevalues of A with maximal and minimal magnitudes respectively. All orthogonal, symmetric, and skew-symmetric matrices are normal. The condition number measures how much the columns/rows of a matrix are dependent on each other; higher the value of the condition number, more is the linear dependence. Condition number 1 means that the columns/rows of a matrix are linearly independent.

```
Set $\mathbf{q}_1 = \frac{1}{||\mathbf{b}||}\mathbf{b}$. //The first step in Gram schmidt.
for $j = 1$ to $n-1$ do
  $\mathbf{t} = A\mathbf{q}_j$.
  for $i = 1$ to $j$ do
    //If $A$ is symmetric, it will be $i = \max(1, j-1)$ to $j$.
    $H_{i,j} = \mathbf{q}_i^T\mathbf{t}$.
    $\mathbf{t} = \mathbf{t} - H_{i,j}\mathbf{q}_i$.
  end for
  $H_{j+1,j} = ||\mathbf{t}||$.
  $\mathbf{q}_{j+1} = \frac{1}{||\mathbf{t}||}\mathbf{t}$.
end for
$\mathbf{t} = A\mathbf{q}_n$.
for $i = 1$ to $n$ do
  //If $A$ is symmetric, it will be $i = n-1$ to $n$.
  $H_{i,n} = \mathbf{q}_i^T\mathbf{t}$.
  $\mathbf{t} = \mathbf{t} - H_{in}\mathbf{q}_i$.
end for
$H_{j+1,j} = ||\mathbf{t}||$.
$\mathbf{q}_{j+1} = \frac{1}{||\mathbf{t}||}\mathbf{t}$.
```

Figure 3.51: The Arnoldi algorithm for computing orthonormal basis.

and the vector $\mathbf{b}$

$$\mathbf{b} = \begin{bmatrix} 0.6382 & 0.3656 & 0.1124 & 0.5317 \end{bmatrix}^T$$

The matrix K_4 is

$$K_4 = \begin{bmatrix} 0.6382 & 1.8074 & 8.1892 & 34.6516 \\ 0.3656 & 1.7126 & 7.5403 & 32.7065 \\ 0.1124 & 1.7019 & 7.4070 & 31.9708 \\ 0.5317 & 1.9908 & 7.9822 & 34.8840 \end{bmatrix}$$

Its condition number is 1080.4.

The algorithm in Figure 3.51 computed the following basis for the matrix K_4.

$$Q_4 = \begin{bmatrix} 0.6979 & -0.3493 & 0.5101 & -0.3616 \\ 0.3998 & 0.2688 & 0.2354 & 0.8441 \\ 0.1229 & 0.8965 & 0.1687 & -0.3908 \\ 0.5814 & 0.0449 & -0.8099 & -0.0638 \end{bmatrix}$$

The coefficient matrix H_4 is

$$H_4 = \begin{bmatrix} 3.6226 & 1.5793 & 0 & 0 \\ 1.5793 & 0.6466 & 0.5108 & 0 \\ 0 & 0.5108 & -0.8548 & 0.4869 \\ 0 & 0 & 0.4869 & 0.3459 \end{bmatrix}$$

and its eigenvalues are 4.3125, 0.5677, -1.2035 and 0.0835. On the other hand, the following matrix H_3 (obtained by restricting to K_3) has eigenvalues 4.3124, 0.1760 and -1.0741.

The basic conjugate gradient method selects vectors in $\mathbf{x}_k \in \mathcal{K}_k$ that approach the exact solution to $A\mathbf{x} = \mathbf{b}$. Following are the main ideas in the conjugate gradient method.

1. The rule is to select an $\mathbf{x}_k$ so that the new residual $\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$ is orthogonal to all the previous residuals. Since $A\mathbf{x}_k \in \mathcal{K}_{k+1}$, we must have $\mathbf{r}_k \in \mathcal{K}_{k+1}$ and $\mathbf{r}_k$ must be orthogonal to all vectors in $\mathcal{K}_k$. Thus, $\mathbf{r}_k$ must be a multiple of $\mathbf{q}_{k+1}$. This holds for all k and implies that
$$\mathbf{r}_k^T\mathbf{r}_i = 0$$
for all $i < k$.

2. Consequently, the difference $\mathbf{r}_k - \mathbf{r}_{k-1}$, which is a linear combination of $\mathbf{q}_{k+1}$ and $\mathbf{q}_k$, is orthogonal to each subspace $\mathcal{K}_i$ for $i < k$.

3. Now, $\mathbf{x}_i - \mathbf{x}_{i-1}$ lies in the subspace $\mathcal{K}_i$. Thus, $\Delta\mathbf{r} = \mathbf{r}_k - \mathbf{r}_{k-1}$ is orthogonal to all the previous $\Delta\mathbf{x} = \mathbf{x}_i - \mathbf{x}_{i-1}$. Since $\mathbf{r}_k - \mathbf{r}_{k-1} = -A(\mathbf{x}_k - \mathbf{x}_{k-1})$, we get the following 'conjugate directions' condition for the updates
$$(\mathbf{x}_i - \mathbf{x}_{i-1})^T A(\mathbf{x}_k - \mathbf{x}_{k-1}) = 0$$
for all $i < k$. This is a necessary and sufficient condition for the orthogonality of the new residual to all the previous residuals. Note that while the residual updates are orthogonal in the usual inner product, the variable updates are orthogonal in the inner product with respect to A.

The basic conjugate gradient method consists of 5 steps. Each iteration of the algorithm involves a multiplication of vector $\mathbf{d}_{k-1}$ by A and computation of two inner products. In addition, an iteration also involves around three vector updates. So each iteration should take time upto $(2+\theta)n$, where θ is determined by the sparsity of matrix A. The error $\mathbf{e}_k$ after k iterations is bounded as follows.

$$||\mathbf{e}_k||_A = (\mathbf{x}_k - \mathbf{x})^T A(\mathbf{x}_k - \mathbf{x}) \leq 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k ||\mathbf{e}_0||$$

The 'gradient' part of the name *conjugate gradient* stems from the fact that solving the linear system $A\mathbf{x} = \mathbf{b}$ is corresponds to finding the minimum value

```
$\mathbf{x}_0 = \mathbf{0}$, $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{d}_0 = \mathbf{r}_0$, $k = 1$.
**repeat**
    1. $\alpha_k = \frac{\mathbf{r}_{k-1}^T\mathbf{r}_{k-1}}{\mathbf{d}_{k-1}^T A\mathbf{d}_{k-1}}$. //Step length for next update. This corresponds to
    the entry $H_{k,k}$.
    2. $\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{d}_{k-1}$.
    3. $\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k A\mathbf{d}_{k-1}$. //New residual obtained using $\mathbf{r}_k - \mathbf{r}_{k-1} =$
    $-A(\mathbf{x}_k - \mathbf{x}_{k-1})$.
    4. $\beta_k = \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{r}_{k-1}^T\mathbf{r}_{k-1}}$. //Improvement over previous step. This corresponds
    to the entry $H_{k,k+1}$.
    5. $\mathbf{d}_k = \mathbf{r}_k + \beta_k \mathbf{d}_{k-1}$. //The next search direction, which should be
    orthogonal to the search direction just used.
    $k = k + 1$.
**until** $\beta_k < \theta$.
```

Figure 3.52: The conjugate gradient algorithm for solving $A\mathbf{x} = \mathbf{b}$ or equivalently, for minimizing $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$.

of the convex (for positive definite A) energy function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} = \mathbf{r}$ by setting its gradient $A\mathbf{x} - \mathbf{b}$ to the zero vector. The steepest descent method makes a move along at the direction of the residual $\mathbf{r}$ at every step but it does not have a great convergence; we land up doing a lot of work to make a little progress. In contrast, as reflect in the step $\mathbf{d}_k = \mathbf{r}_k + \beta_k\mathbf{d}_{k-1}$, the conjugate gradient method makes a step in the direction of the residual, but only after removing any component β_k along the direction of the step it just took. Figures 3.53 and 3.54 depict the steps taken by the steepest descent and the conjugate descent techniques respectively, on the level-curves of the function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$, in two dimensions. It can be seen that while the steepest descent technique requires many iterations for convergence, owing to its oscillations, the conjugate gradient method takes steps that are orthogonal with respect to A (or are orthogonal in the transfomed space obtained by multiplying with A), thus taking into account the geometry of the problem and taking a fewer number of steps. If the matrix A is a hessian, the steps taken by conjugate gradient are orthogonal in the local Mahalonobis metric induced by the curvature matrix A. Note that if $\mathbf{x}^{(0)} = \mathbf{0}$, the first step taken by both methods will be the same.

The conjugate gradient method is guaranteed to reach the minimum of the energy function E in exactly n steps. Further, if A has only r distinct eigenvalues, then the conjugate gradient method will terminate at the solution in at most r iterations.

3.5.8 Conjugate Gradient

We have seen that the Conjugate Gradient method in Figure 3.52 can be viewed as a minimization algorithm for the convex quadratic function $E(\mathbf{x}) =$

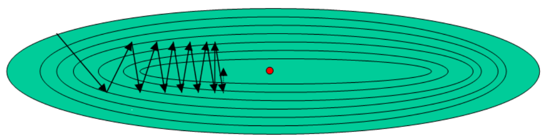


Figure 3.53: Illustration of the steepest descent technique on level curves of the function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$.

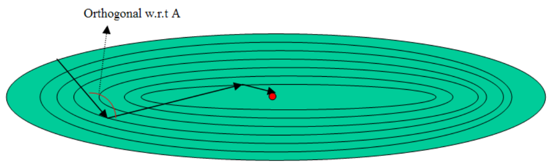


Figure 3.54: Illustration of the conjugate gradient technique on level curves of the function $E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$.

$\frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$. Can the approach be adapted to minimize general nonlinear convex functions? Nonlinear variants of the conjugate gradient are well studied [**?**] and have proved to be quite successful in practice. The general conjugate gradient method is essentially an incremental way of doing second order search.

Fletcher and Reeves showed how to extend the conjugate gradient method to nonlinear functions by making two simple changes[33] to the algorithm in Figure 3.52. First, in place of the exact line search formula in step (1) for the step length α_k, we need to perform a line search that identifies an approximate minimum of the nonlinear function f along $\mathbf{d}^{(k-1)}$. Second, the residual $\mathbf{r}^{(k)}$, which is simply the gradient of E (and which points in the direction of decreasing value of E), must be replaced by the gradient of the nonlinear objective f, which serves a similar purpose. These changes give rise to the algorithm for nonlinear optimization outlined in Figure 3.55. The search directions $\mathbf{d}^{(k)}$ are computed by Gram-Schmidt conjugation of the residuals as with linear conjugate gradient. The algorithm is very sensitive to the line minimization step and it generally requires a very good line minimization. Any line search procedure that yields an α_k satisfying the strong Wolfe conditions (see (3.90) and (3.91)) will ensure that all directions $\mathbf{d}^{(k)}$ are descent directions for the function f, otherwise, $\mathbf{d}^{(k)}$ may cease to remian a descent direction as iterations proceed. We note that each iteration of this method costs on $O(n)$, as against the Newton or quasi-newton methods which cost atleast $O(n^2)$ owing to matrix operations. Most often, it yields optimal progress after $h << n$ iterations. Due to this property, the conjugate gradient method drives nearly all large-scale optimization today.

[33]We note that in the algorithm in Figure 3.52, the residuals $\mathbf{r}^{(k)}$ in successive iterations (which are gradients of E) are orthogonal to each other, while the corresponding update directions are orthogonal with respect to A. While the former property is difficult to enforce for general non-linear functions, the latter condition can be enforced.

```
Select $\mathbf{x}^{(0)}$, Let $f_0 = f(\mathbf{x}^{(0)})$, $\mathbf{f}_0 = \nabla f(\mathbf{x}^{(0)})$, $\mathbf{d}^{(0)} = -\nabla \mathbf{f}_0$, $k = 1$.
repeat
  1. Compute $\alpha_k$ by line search.
  2. Set $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \alpha_k \mathbf{d}^{(k-1)}$.
  3. Evaluate $\mathbf{f}^{(k)} = \nabla f(\mathbf{x}^{(k)})$.
  4. $\beta_k = \frac{\left(\mathbf{f}^{(k)}\right)^T \mathbf{f}^{(k)}}{\left(\mathbf{f}^{(k-1)}\right)^T \mathbf{f}^{(k-1)}}$.
  5. $\mathbf{d}_k = -\mathbf{f}^{(k)} + \beta_k \mathbf{d}^{(k-1)}$.
  $k = k + 1$.
until $\frac{||\mathbf{f}^{(k)}||}{\mathbf{f}^{(0)}} < \theta$ OR $k > maxIter$.
```

Figure 3.55: The conjugate gradient algorithm for optimizing nonlinear convex function f.

It revolutionalized optimization ever since it was invented in 1954.

Variants of the Fletcher-Reeves method use different choices of the parameter β_k. An important variant, proposed by Polak and Ribiere, defines β_k as

$$\beta_k^{PR} = \frac{\left(\mathbf{f}^{(k)}\right)^T \left(\mathbf{f}^{(k)} - \mathbf{f}^{(k-1)}\right)}{\left(\mathbf{f}^{(k)}\right)^T \mathbf{f}^{(k)}}$$

The Fletcher-Reeves method converges if the starting point is sufficiently close to the desired minimum. However, convergence of the Polak-Ribiere method can be guaranteed by choosing

$$\beta_k = max\left\{\beta_k^{PR}, 0\right\}$$

Using this value is equivalent to restarting[34] conjugate gradient if $\beta_k^{PR} < 0$. In practice, the Polak-Ribiere method converges much more quickly than the Fletcher-Reeves method. It is generally required to restart the conjugate gradient method after every n iterations, in order to get back conjugacy, *etc.*

If we choose f to be the strongly convex quadratic E and α_k to be the exact minimizer, this algorithm reduces to the linear conjugate gradient method, Unlike the linear conjugate gradient method, whose convergence properties are well understood and which is known to be optimal (see page 265), nonlinear conjugate gradient methods sometimes show bizarre convergence properties. It has been proved by Al-Baali that if the level set $\mathcal{L} = \left\{\mathbf{x}|f(\mathbf{x}) \leq f(\mathbf{x}^{(0)}\right\}$ of a convex function f is bounded and in some open neighborbood of $\mathcal{L}$, f is Lipshitz continuously differentiable and that the algorithm is implemented with a line search that satisfies the strong Wolfe conditions, with $0 < c_1 < c_2 < 1$, then

$$\lim_{k\to\infty} \inf \; ||\mathbf{f}^{(k)}|| = 0$$

[34]Restarting conjugate gradient means forgetting the past search directions, and start it anew in the direction of steepest descent.

In summary, quasi-Newton methods are robust. But, they require $O(n^2)$ memory space to store the approximate Hessian inverse, and so they are not directly suited for large scale problems. Modificationsof these methods called Limited Memory Quasi-Newton methods use $O(n)$ memory and they are suited for large scale problems. Conjugate gradient methods also work well and are well suited for large scale problems. However they need to be implemented carefully, with a carefully set line search. In some situations block coordinate descent methods (optimizing a selected subset of variables at a time) can be very much better suited than the above methods.

3.6 Algorithms for Constrained Minimization

The general form of constrained convex optimization problem was given in (3.20) and is restated below.

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array} \tag{3.99}$$

For example, when f is linear and g_i's are polyhedral, the problem is a linear program, which was stated in (3.83) and whose dual was discussed on page 233. Linear programming is a typical example of constraint minimization problem and will form the subject matter for discussion in Section 3.7. As another example, when f is quadratic (of the form $\mathbf{x}^T Q\mathbf{x} + \mathbf{b}^T\mathbf{x}$) and g_i's are polyhedral, the problem is called a quadratic programming problem. A special case of quadratic programming is the least squares problem, which we will take up in details in Section 3.8.

3.6.1 Equality Constrained Minimization

The simpler form of constrained convex optimization is when there is only the equality constrained in problem (3.99) and it turns out to be not much different from the unconstrained case. The equality constrained convex problem can be more explicitly stated as in (3.100).

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array} \tag{3.100}$$

where f is a convex and twice continuously differentiable function and $A \in \Re^{p \times n}$ has rank p. We will assume that the finite primal optimal value p^* is attained by f at some point $\widehat{\mathbf{x}}$. The following fundamental theorem for the equality constrained convex problem (3.100) can be derived using the KKT conditions stated in Section 3.4.4 that were proved to be necessary and sufficiency conditions for optimality of a convex problem with differentible objective and constraint functions.

Theorem 70 $\widehat{\mathbf{x}}$ *is optimal point for the primal iff there exists a* $\widehat{\mu}$ *such that the following conditions are satisfied.*

$$\begin{aligned} &\nabla f(\widehat{\mathbf{x}}) + A^T\widehat{\mu} = \mathbf{0} \\ &A\widehat{\mathbf{x}} = \mathbf{b} \end{aligned} \tag{3.101}$$

The term $\nabla f(\widehat{\mathbf{x}}) + A^T\widehat{\mu}$ *is sometimes called the dual residual* (r_d) *while the term* $A\widehat{\mathbf{x}} - \mathbf{b}$ *is referred to as the primal residual* (r_p). *The optimality condition basically states that both* r_d *and* r_p *should both be* 0 *and the the success of this test is a certificate of optimality.*

As an illustration of this theorem, consider the constrained quadratic problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + c \\ \text{subject to} & P\mathbf{x} = \mathbf{q} \end{array} \tag{3.102}$$

By theorem 70, the necessary and sufficient condition for optimality of a point $(\widehat{\mathbf{x}}, \widehat{\lambda})$ is

$$\underbrace{\begin{bmatrix} A & P^T \\ P & 0 \end{bmatrix}}_{KKT\ matrix} \begin{bmatrix} \widehat{\mathbf{x}} \\ \widehat{\lambda} \end{bmatrix} = \begin{bmatrix} -\mathbf{b} \\ \mathbf{q} \end{bmatrix}$$

The KKT matrix[35] is nonsingular *iff*, $P + A^T A \succ 0$. In such an event, the system of $n+p$ linear equations in $n+p$ unknowns will have a unique solution corresponding to the point of global minimum of (3.102). The linearly constrained least squared problem is a specific example of this and is discussed in Section 3.8.2.

Eliminating Equality Constraints

Figure 2.3 summarized the number of solutions to the system $A\mathbf{x} = \mathbf{b}$ under different conditions. In particular, when the rank of A is the number of its rows (p) and is less than the number of its columns (n), there are infinitely many solutions. This was logically derived in (2.35), and we restate it here for reference:

$$\mathbf{x}_{complete} = \mathbf{x}_{particular} + \mathbf{x}_{nullspace}$$

where the three vectors are defined with respect to the reduced row echelon form R of A (*c.f.* Section 2.6.2):

[35]This matrix comes up very often in many areas such as optimization, mechanics, *etc.*

1. $\mathbf{x}_{complete}$: specifies any solution to $A\mathbf{x} = \mathbf{b}$
2. $\mathbf{x}_{particular}$: is obtained by setting all free variables (corresponding to columns with no pivots) to 0 and solving $A\mathbf{x} = \mathbf{b}$ for pivot variables.
3. $\mathbf{x}_{nullspace}$: is any vector in the null space of the matrix A, obtained as a linear combination of the basis vectors for $N(A)$.

Using formula (2.27) on page 113 to derive the null basis $N \in \Re^{n \times n-p}$ (that is, $AN = 0$ and the columns of N span $N(A)$), we get the following free parameter expression for the solution set to $A\mathbf{x} = \mathbf{b}$:

$$\{\mathbf{x} \,|A\mathbf{x} = \mathbf{b}\} = \left\{N\mathbf{z} + \mathbf{x}_{particular} \left|\mathbf{z} \in \Re^{n-p}\right.\right\}$$

We can express the constrained problem in (3.100) in terms of the variables $\mathbf{z} \in \Re^{n-p}$ (that is through an affine change of coordinates) to get the following equivalent problem:

$$\underset{\mathbf{z} \in \Re^{n-1}}{\text{minimize}} \quad f(N\mathbf{z} + \mathbf{x}_{particular}) \tag{3.103}$$

This problem is equivalent to the original problem in (3.100), has no equality constraints and has p fewer variables. The optimal solutions $\widehat{\mathbf{x}}$ and $\widehat{\mu}$ to the primal and dual of (3.100) respectively can be expressed in terms of the optimal solution $\widehat{\mathbf{z}}$ to (**??**) as:

$$\begin{aligned} \widehat{\mathbf{x}} &= N\widehat{\mathbf{z}} + \mathbf{x}_{particular} \\ \widehat{\mu} &= -(AA^T)^{-1}A\nabla f(\widehat{\mathbf{x}}) \end{aligned} \tag{3.104}$$

Any iterative algorithm that is applied to solve the problem (3.104) will ensure that all intermediate points are feasible, since for any $\mathbf{z} \in \Re^{n-p}$, $\mathbf{x} = F\mathbf{z} + \mathbf{x}_{particular}$ is feasible, that is, $A\mathbf{x} = \mathbf{b}$. However, when the Newton's method is applied, the iterates are independent of the exact affine change of coordinates induced by the choice of the null basis F (*c.f.* page 250). The Newton update rule $\Delta\mathbf{z}^{(k)}$ for (3.103) is given by the solution to:

$$N\nabla^2 f(N\mathbf{z}^{(k)} + \mathbf{x}_{particular})N^T\Delta\mathbf{z}^{(k)} = N\nabla f(N\mathbf{z}^{(k)} + \mathbf{x}_{particular})$$

Due the affine invariance of Newton's method, if $\mathbf{z}^{(0)}$ is the starting iterate and $\mathbf{x}^{(0)} = N\mathbf{z}^{(0)} + \mathbf{x}_{particular}$, the k^{th} iterate $\mathbf{x}^{(k)} = N\mathbf{z}^{(k)} + \mathbf{x}_{particular}$ is independent of the choice of the null basis N. We therefore do not need seperate convergence analysis. The algorithm for the Newton's method was outlined in Figure 3.49. Techniques for handling constrained optimization using Newton's method given an infeasible starting point $\mathbf{x}^{(0)}$ can be found in [**?**].

3.6.2 Inequality Constrained Minimization

The general inequality constrained convex minimization problem is

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & g_i(\mathbf{x}) \leq \mathbf{0}, \quad i = 1, \ldots, m \\ & A\mathbf{x} = b \end{array} \tag{3.105}$$

where f as well as the g_i's are convex and twice continuously differentiable. As in the case of equality constrained optimization, we will assume that $A \in \Re^{p \times n}$ and has rank p. Further, we will also assume that the finite primal optimal value p^* is attained by f at some point $\widehat{\mathbf{x}}$. Finally, we will assume that the Slaters constraint qualification (*c.f.* page 236) conditions hold so that strong duality holds and the dual optimum is attained. Linear programs (LP), quadratically constrained quadratic programs (QCQP) (all listed in table 3.4 on page 236) and geometric programs[36] (GP) are some examples of convex optimization problems with inequality constraints. An example geometric program (in its convex form) is

$$\begin{array}{llr} \underset{\mathbf{y} \in \Re^n}{\text{minimize}} & \log\left(\sum_{k=1}^{q} e^{\mathbf{a}_k^T \mathbf{y} + b_k}\right) & \\ \text{subject to} & \log\left(\sum_{k=1}^{r} e^{\mathbf{c}_k^T \mathbf{y} + d_k}\right) \leq 0 & i = 1, 2, \ldots, p \\ & \mathbf{g}_i^T \mathbf{y} + h_i & i = 1, 2, \ldots, m \end{array} \tag{3.106}$$

Semi-definite programs (SDPs) do not satisfy conditions such as zero duality gap, *etc.*, but can be handled by extensions of interior-point methods to problems having generalized inequalities.

Logarithmic Barrier

One idea for solving a minimization problem with inequalities is to replace the inequalities by a so-called barrier term. The barrier term is subtracted from the objective function with a weight μ on it. The solution to (3.105) is approximated by the solution to the following problem.

$$\begin{array}{ll} \text{minimize} & B(\mathbf{x}, \mu) = f(\mathbf{x}) - \mu \sum_{i=1}^{m} \ln\left(-g_i(\mathbf{x})\right) \\ \text{subject to} & A\mathbf{x} = b \end{array} \tag{3.107}$$

[36]Although geometric programs are not convex in their natural form, they can, however, be transformed to convex optimization problems, by a change of variables and a transformation of the objective and constraint functions.

The objective function $B(\mathbf{x}, \mu)$ is called the *logarithmic barrier function.* This function is convex, which can be proved by invoking the composition rules described in Section 3.2.10. It is also twice continuously differentiable. The barrier term, as a function of $\mathbf{x}$ approaches $+\infty$ as any feasible interior point $\mathbf{x}$ approaches the boundary of the feasible region. Because we are minimizing, this property prevents the feasible iterates from crossing the boundary and becoming infeasible. We will denote the point of optimality $\widehat{\mathbf{x}}(\mu)$ as a function of μ.

However, the optimal solution to the original problem (a typical example being the LP discussed in Section 3.7) is typically a point on the boundary of the feasible region (we will see this in the case of linear programming in Section 3.7). To obtain such a boundary point solution, it is necessary to keep decreasing the parameter μ of the barrier function to 0 in the limit. As a very simple example, consider the following inequality constrained optimization problem.

$$\begin{array}{ll} \text{minimize} & x^2 \\ \text{subject to} & x \geq 1 \end{array}$$

The logarithmic barrier formulation of this problem is

$$\text{minimize} \quad x^2 - \mu \ln (x - 1)$$

The unconstrained minimizer for this convex logarithmic barrier function is $\widehat{\mathbf{x}}(\mu) = \frac{1}{2} + \frac{1}{2}\sqrt{1 + 2\mu}$. As $\mu \to 0$, the optimal point of the logarithmic barrier problem approaches the actual point of optimality $\widehat{\mathbf{x}} = 1$ (which, as we can see, lies on the boundary of the feasible region). The generalized idea, that as $\mu \to 0$, $f(\widehat{\mathbf{x}}) \to p^*$ (where p^* is the optimal for (3.105)) will be proved next.

Properties of the estimate $f(\widehat{\mathbf{x}}(\mu))$

The following are necessary and sufficient conditions for $\widehat{\mathbf{x}}(\mu)$ to be a solution to (3.107) for a fixed μ (see KKT conditions in (3.88)):

1. The point $\widehat{\mathbf{x}}(\mu)$ must be strictly feasible. That is,

$$A\widehat{\mathbf{x}}(\mu) = \mathbf{b}$$

   and

$$g_i(\widehat{\mathbf{x}}(\mu)) < 0$$

2. There must exist a $\eta \in \Re^p$ such that

$$\nabla f(\widehat{\mathbf{x}}(\mu)) + \sum_{i=1}^{m} \frac{-\mu}{g_i(\widehat{\mathbf{x}}(\mu))} \nabla g_i(\widehat{\mathbf{x}}(\mu)) + A^T \widehat{\eta} = \mathbf{0} \qquad (3.108)$$

Define

$$\widehat{\lambda}_i(\mu) = \frac{-\mu}{g_i(\widehat{\mathbf{x}}(\mu))}$$

and

$$\widehat{\eta}(\mu) = \widehat{\eta}\mu$$

We claim that the pair $(\widehat{\lambda}(\mu), \widehat{\eta}(\mu))$ is dual feasible. The following steps prove our claim

1. Since $g_i(\widehat{\mathbf{x}}(\mu)) < 0$ for $i = 1, 2, \dots, m$, $\widehat{\lambda}(\mu) \succ \mathbf{0}$.

2. Based on the proof of theorem 67, we can infer that $L(\mathbf{x}, \lambda, \eta)$ is convex in $\mathbf{x}$.

$$L(\mathbf{x}, \lambda, \eta) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \eta^T (A\mathbf{x} - \mathbf{b})$$

   Since the lagrangian is convex in $\mathbf{x}$ and since it is differentiable on its domain, from (3.108), we can conclude that $\widehat{\mathbf{x}}(\mu)$ is a critical point of $L(\mathbf{x}, \lambda, \eta)$ and therefore minimizes it for $(\widehat{\lambda}(\mu), \widehat{\eta}(\mu))$.

3. That is, the dual $L^*(\widehat{\lambda}(\mu), \widehat{\eta}(\mu))$ is defined and therefore, $(\widehat{\lambda}(\mu), \widehat{\eta}(\mu))$ is dual feasible.

$$L^*(\widehat{\lambda}(\mu), \widehat{\eta}(\mu)) = f(\widehat{\mathbf{x}}(\mu)) + \sum_{i=1}^{m} \widehat{\lambda}_i g_i(\widehat{\mathbf{x}}(\mu)) + \widehat{\eta}(\mu)^T (A\widehat{\mathbf{x}}(\mu) - \mathbf{b}) = f(\widehat{\mathbf{x}}(\mu)) - m\mu \tag{3.109}$$

From the weak duality theorem 66, we know that $d^* \le p^*$, where d^* and p^* are the primal and dual optimals respectively, for (3.105). Since $L^*(\widehat{\lambda}(\mu), \widehat{\eta}(\mu)) \le d^*$ (by definition), we will have from (3.109), $f(\widehat{\mathbf{x}}(\mu)) - m\mu \le p^*$. Or equivalently,

$$f(\widehat{\mathbf{x}}(\mu)) - p^* \le m\mu \tag{3.110}$$

The inequality in (3.110) forms the basis of the barrier method; it confirms the intuitive idea that $\widehat{\mathbf{x}}(\mu)$ converges to an optimal point as $\mu \to 0$. We will next discuss the barrier method.

The Barrier Method

The barrier method is a simple extension of the unconstrained minimization method to inequality constrained minimization. This method is based on the property in (3.110). This method solves a sequence of unconstrained (or linearly constrained) minimization problems, using the last point found as the starting point for the next unconstrained minimization problem. It computes $\widehat{\mathbf{x}}(\mu)$ for a sequence of decreasing values of μ, until $m\mu \leq \epsilon$, which guarantees that we have an ϵ-suboptimal solution of the original problem. It was originally proposed as the sequential unconstrained minimization technique (SUMT) technique by Fiacco and McCormick in the 1960s. A simple version of the method is outlined in Figure 3.56.

```
Find a strictly feasible starting point $\widehat{\mathbf{x}}$, $\mu = \mu^{(0)} > 0$, $\alpha > 0$.
Select an appropriate tolerance $\epsilon > 0$.
repeat
    1. Centering Step: Compute $\widehat{\mathbf{x}}(\mu)$ by minimizing $B(\mathbf{x}, \mu)$ (optionally
    subject to $A\mathbf{x} = \mathbf{b}$) starting at $\mathbf{x}$.
    2. Update $\mathbf{x} = \widehat{\mathbf{x}}(\mu)$.
    3. If $m\mu \leq \epsilon$, quit.
    4. Decrease $\mu$: $\mu = \alpha\mu$.
until
```

Figure 3.56: The Barrier method.

The centering step (1) can be executed using any of the descent techniques discussed in Section 3.5. It can be proved [?] that the duality gap is $m\mu^{(0)}\alpha^k$ after k iterations. Therefore, the desired accuracy ϵ can be achieved by the barrier method after exactly $\left\lceil \frac{\log\left(\frac{m\mu^{(0)}}{\epsilon}\right)}{-log(\alpha)} \right\rceil$ steps.

Successive minima $\widehat{\mathbf{x}}(\mu)$ of the Barrier function $B(\mathbf{x}, \mu)$ can be shown to have the following properties. Let $\overline{\mu} < \mu$ for sufficiently small μ, then

1. $B(\widehat{\mathbf{x}}(\overline{\mu}), \overline{\mu}) < B(\widehat{\mathbf{x}}(\mu), \mu)$
2. $f(\widehat{\mathbf{x}}(\overline{\mu})) \leq f(\widehat{\mathbf{x}}(\mu))$
3. $-\sum_{i=1}^{m} \ln\left(-g_i(\widehat{\mathbf{x}}(\overline{\mu}))\right) \geq -\sum_{i=1}^{m} \ln\left(-g_i(\widehat{\mathbf{x}}(\mu))\right)$

When a strictly feasible point $\widehat{\mathbf{x}}$ is not known, the barrier method is preceded by a preliminary stage, called phase I, in which a strictly feasible point is computed (if it exists). The strictly feasible point found during phase I is then used as the starting point for the barrier method. This is discussed in greater details in [?].

3.7 Linear Programming

Linear programming has been widely used in the industry for maximizing profits, minimizing costs, *etc.* The word *linear* implies that the cost function is linear in the form of an inner product.

The inputs to the program are

1. $\mathbf{c}$, a cost vector of size n.

2. An $m \times n$ matrix A.

3. A vector $\mathbf{b}$ of size m.

The unknown is a vector $\mathbf{x}$ of size n, and this is what we will try to determine.

In linear programming (LP), the task is to minimize a linear objective function of the form $\sum_{j=1}^{n} c_j x_j$, subject to linear inequality constraints[37] of the form $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$, $i = 1, \ldots, m$ and $x_i > 0$. The problem can be stated as in (3.111). In contrast to the LP specification on page 233, where the constraint $\mathbf{x} \geq \mathbf{0}$ was absorbed into the more general constraint $-A\mathbf{x} + \mathbf{b} \leq \mathbf{0}$, here we choose to specify it as a seperate constraint.

$$\begin{array}{ll} \min\limits_{\mathbf{x} \in \Re^n} & \mathbf{x}^T \mathbf{c} \\ \text{subject to} & -A\mathbf{x} + \mathbf{b} \leq \mathbf{0} \quad \mathbf{x} \geq \mathbf{0} \end{array} \tag{3.111}$$

The flip side of this problem is that it has no analytical formula as a solution. However, that does not make a big difference in practice, because there exist reliable and efficient algorithms and software for linear programming. The computational time is roughly proportional to $n^2 m$, if $m \geq n$. This is basically the cost of one iteration in an interior point method.

Linear programming (LP) problems are harder to recognize in practice and often need reformulations to get into the standard form in (3.111). Minimizing a piecewise linear function of x is not an LP, thought it can be written and solved as an LP. Other problems involving 1 or ∞ norms can also be written as linear programming problems.

The basis for linear programing was mentioned on page 194; linear functions have no critical points and therefore, by theorem 45, the extreme values are always assumed at the boundary of the feasible set. In the case of linear programs, the feasible set is itself defined by linear inequalities: $\{\mathbf{x}| - A\mathbf{x} + \mathbf{b} \leq \mathbf{0}\}$. Applying the argument recursively, it can be proved that the extreme values for a linear program are assumed at some corners (*i.e.*, vertices) of the feasible set. A *corner* is the intersection point of n different planes, each given by a single equation. That is, a corner point is obtained by turning n of the $n+m$ inequalities into equalities and finding their intersection[38] An edge is the intersection of $n-1$ inequalities and connects two corners. Geometrically, it can be observed that when you maximize or minimize some linear function, as your progress in one direction in the search space, the objective will either increase monotonically or decrease monotonically. Therefore, the maximum and minimum will be found at the corners of the allowed region.

[37] It is a rare feature to have linear inequality constraints.

[38] In general, there are $\frac{(n+m)!}{n!m!}$ intersections.

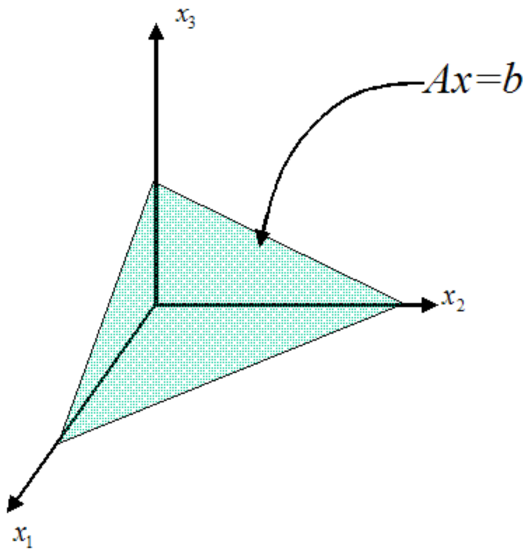


Figure 3.57: Example of the feasible set of a linear program for $n = 3$.

The feasible set is in the form of a finite interval in n dimensions. Figure 3.57 pictorially depicts a typical example of the feasible region for $n = 3$. The constraints $A\mathbf{x} \geq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$ would allow a tetrahedron or pyramid in the first (or completely positive) octant. If the constraint was an equality, $A\mathbf{x} = \mathbf{b}$, the feasible set would be the shaded traingle in the figure. In general for any n, the constraint $A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ would yield as the feasible set, a polyhedron. The task of maximizing (or miminizing) the linear objective function $\mathbf{x}^T\mathbf{c} = \sum_{i=1}^{n} c_i x_i$ translates to finding a solution at one of the corners of the feasible region. Corners are points where some of the inequality constraints are tight or active, and others are not. At the corners, some of the inequality constraints translate to equalities. It is just a question of finding the right corner.

Why not just search all corners for the optimal answer? The trouble is that there are lots of corners. In n dimensions, with m constraints, the number of corners grows exponentially and there is no way to check all of them. There is an interesting competition between two quite different approaches for solving linear programs:

1. *The simplex method*

2. *Interior point barrier method*

3.7.1 Simplex Method

The simplex algorithm [**?**] is one of the fundamenetal methods for linear programming, developed in the late 1940s by Dantzig. It is the best established approach for solving linear problems. In the worst case, the algorithm takes a number of steps that is exponential in n; but, in practice it is the most efficient method for solving linear programs.

The simplex method first constructs an admissible solution at a corner (which can be quite a bit of a job) of the polyhedron and then moves along its edges to vertices with successively higher values of the objective function until the optimum is reached. The movement along an edge originating at a vertex is performed by 'loosening' one of the inequalities that were tight at the vertex. The inequality chosen for 'loosening' is the one promising the fastest drop in the objective function $\mathbf{x}^T\mathbf{c}$. The rate of decrease along an edge can be measured using the gradient of the objective. This procedure is carried out iteratively, till the method encounters a vertex which has no edge (constraint) that is a promising descent direction (which means that the cost goes up along all edges incident at that vertex). Since an edge corresponding to decreasing value of the objective cannot correspond to its increasing value, no edge will be traversed twice in this process.

We will first rewrite the constraints $A\mathbf{x} \geq \mathbf{b}$ in the above LP as equations, by introducing a new non-negative "slack" variable s_j for the j^{th} constraint (for all j's) and subtracting it from the left-hand side of each inequality:

$$A\mathbf{x} - \mathbf{s} = \mathbf{b}$$

or equivalently in matrix notation

$$[-A \ \ +I] \left[\ \mathbf{x} \ \mathbf{s} \ \right] = -\mathbf{b}$$

We will treat the $m \times n + m$ matrix $M = [-A \ \ I]$ as our new coefficient matrix and $\mathbf{y} = [\mathbf{x} \ \ \mathbf{s}]^T$ as our new variable vector. With this, the above constraint can be rewritten as

$$M\mathbf{y} = -\mathbf{b}$$

The feasible set is now governed by these m equality constraints and the $n + m$ non-negativity constraints $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{y} \geq \mathbf{0}$. The original cost vector $\mathbf{c}$ is extended to a vector $\mathbf{d}$ by appending m more zero components. This leaves us with the following problem, equivalent to the original LP (3.111).

$$\begin{array}{ll} \min\limits_{\mathbf{y} \in \Re^{n+m}} & \mathbf{y}^T\mathbf{d} \\ \text{subject to} & M\mathbf{y} = -\mathbf{b} \quad \mathbf{y} \geq \mathbf{0} \end{array} \tag{3.112}$$

We will assume that the matrix A (and therefore M) is of full row rank, that is of rank m. In practice, a preprocessing phase is applied to the user-supplied data to remove some redundancies from the given constraints to get a full row rank matrix.

The following definitions and observations will set the platform for the simplex algorithm, which we will describe subsequently.

1. A vector $\mathbf{y}$ is a *basic feasible point* if it is feasible and if there exists a subset $\mathcal{B}$ of the index set $\{1, 2, ..., n\}$ such that

   (a) $\mathcal{B}$ contains exactly m indices.

   (b) $\mathbf{y} \geq \mathbf{0}$.

   (c) $y_i \geq 0$ can be inactive (that is $y_i > 0$) only if $i \in \mathcal{B}$. In other words, $i \notin \mathcal{B} \Rightarrow y_i = 0$.

   (d) If $\mathbf{m}_i$ is the i^{th} column of M, the $m \times m$ matrix B defined as $B = [\mathbf{m}_i]_{i \in \mathcal{B}}$ is nonsingular.

   A set $\mathcal{B}$ satisfying these properties is called a *basis* for the problem (3.112). The corresponding matrix B is called the basis matrix. Any variable y_i for $i \in \mathcal{B}$ is called a *basic variable*, while any variable y_i for $i \notin \mathcal{B}$ is called a *free variable.*

2. It can be seen that all basic feasible points of (3.112) are corners of the feasible simplex $\mathcal{S} = \{\mathbf{x} | A\mathbf{x} \geq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ and vice versa. In other words, a corner of $\mathcal{S}$ corresponds to a point $\mathbf{y}$ in the new representation that has n components as zeroes.

3. Any two corners connected by an edge will have exactly $m - 1$ common basic variables. Each corner has n incident edges (corresponding to the addition of any one of n new basic variables and the corresponding drop of a basic variable).

4. Further, it can be proved that

   (a) If (3.112) has a nonempty feasible region, then there is at least one basic feasible point

   (b) If (3.112) has solutions, then at least one such solution is a basic optimal point

   (c) If (3.112) is feasible and bounded, then it has an optimal solution.

   This is known as the fundamental theorem of linear programming.

Using the ideas and notations presented above, the simplex algorithm can be outlined as follows.

1. Each iterate generated by the simplex algorithm is a *basic feasible point* of (3.112).

2. *Entering free variable:* The next iterate is determined by moving along an edge from one basic feasible solution to another. As discussed above, movement along an edge will mean that $m-1$ variables will remain basic while one will become free. On the other hand, a new free variable will become basic. The real decision is which variable should be removed from the basis and which should be added. The idea in the simplex algorithm is to include that free variable y_k, which has the most negative component d_k (something like steepest descent in the L_1 norm).

3. *Leaving basic variable:* The basic variable from the current basis that will leave next is determined using a *pivot rule*. A commonly applied pivot rule is to determine the leaving basic variable through the constraint (say the j^{th} one) that has the smallest non-negative ratio of the right hand side $b_j^{'}$ of the constraint to the coefficient $m_{jk}^{'}$ of the entering variable y_k. If the coefficients of y_k are negative in all the constraints, it implies an unbounded case; the cost can be made $-\infty$ by arbitrarily increasing the value of y_k.

4. In order to facilitate the easy identification of leaving basic variables, we bring the equations into a form such that the basic variables stand by themselves. This is done by treating the new entering variable y_k as a 'pivot' in the j^{th} equation and substituting its value in terms of the other variables in the j^{th} equation into the other equations (as well as the cost function $\mathbf{y}^T\mathbf{d}$). In this form,

   (a) the protocol is that variables corresponding to all columns of M that are in unit form are basic variables, while the rest are free variables.

   (b) the choice of an equality in the step above automatically entails the choice of the leaving variable - the basic variable y_l corresponding to row j will be the next leaving variable.

5. In a large problem, it is possible for a leaving variable to reenter the basis at a later stage. Unless there is *degeneracy*, the costs keep going down and it can never happen that all of the m basic variables are the same as before. Thus, no corner is revisited and the method must end at the optimal corner or conclude that the cost is unbounded below. Degenracy is said to occur if more than the usual n components of $\mathbf{x}$ are 0 (in which case, cycling might occur but extremely rarely).

Since each simplex step involves decisions (choice of entering and leaving basic variables) and row operations (pivoting *etc.*), it is convenient to fit the data into a large matrix or *tableau*. The operations of the simplex method outlined above can be systematically translated to operations on the tableau.

1. The starting tableau is just a bigger $m+1 \times m+n$ matrix

$$T = \begin{bmatrix} M & -\mathbf{b} \\ \mathbf{d} & \mathbf{0} \end{bmatrix}$$

2. Our first step is to get one basic variable alone on each row. Without loss of generality, we will renumber the variables and rearrange the corresponding coefficients of M so that at every iteration, $y_1, y_2, \ldots y_m$ are the basic variables and the rest are free (*i.e.*, 0). The first m columns of A form an $m \times m$ square matrix B and the last n form an $m \times n$ matrix N. The cost vector $\mathbf{d}$ can also be split as $\mathbf{d}^T = [\mathbf{d}_B^T \;\; \mathbf{d}_N^T]$ and the variable vector can be split as $\mathbf{y}^T = [\mathbf{y}_B^T \;\; \mathbf{y}_N^T]$ with $\mathbf{y}_N = \mathbf{0}$. To operate with the tableau, we will split it as

$$\begin{bmatrix} B & N & -\mathbf{b} \\ \mathbf{d}_B^T & \mathbf{d}_N^T & \mathbf{0} \end{bmatrix}$$

Performing Gauss Jordan elimination on the columns corresponding to basic variables, we get the equations into the form that will be preserved across iterations.

$$\begin{bmatrix} I & B^{-1}N & -B^{-1}\mathbf{b} \\ \mathbf{d}_B^T & \mathbf{d}_N^T & \mathbf{0} \end{bmatrix}$$

Further, we will ensure that all the columns corresponding to basic variables are in the unit form.

$$\begin{bmatrix} I & B^{-1}N & -B^{-1}\mathbf{b} \\ \mathbf{d}_B^T - \mathbf{d}_B^T I = \mathbf{0} & \mathbf{d}_N^T - \mathbf{d}_B^T B^{-1}N & \mathbf{d}_B^T B^{-1}\mathbf{b} \end{bmatrix}$$

This corresponds to a solution $\mathbf{y}_B = -B^{-1}\mathbf{b}$ with cost $\mathbf{d}^T\mathbf{y} = -\mathbf{d}_B^T B^{-1}\mathbf{b}$, which is the negative of the expression on the right hand bottom corner.

3. In the above tableau, the components of the expression $\mathbf{r} = \mathbf{d}_N^T - \mathbf{d}_B^T B^{-1}N$ are the *reduced costs* and capture what it costs to use the existing set of free variables; if the direct cost in $\mathbf{d}_N$ is less than the saving due to use of the other basic variables, it will help to try a free variable. This guides us in the choice of the *entering variable*. If $\mathbf{r} = \mathbf{d}_N^T - \mathbf{d}_B^T B^{-1}N$ has any negative component, then the variable corresponding to the most negative component is picked up as the next entering variable and this choice corresponds to moving from a corner of the polytope $\mathcal{S}$ to an adjacent corner with lower cost. Let y_k be the entering variable and d_k the corresponding cost.

4. As the entering component y_k is increased, to maintain $M\mathbf{y} = -\mathbf{b}$, the first component $\mathbf{y}_j$ that decreases to 0 becomes the leaving variable and transforms from a basic to a free variable. The other components of $\mathbf{y}_B$ would have moved around but would remain positive. Thus, the one that drops to zero should satisfy

$$j = \underset{t=1,2,\ldots,m \;\; (B^{-1}N)_{tk} > 0}{\operatorname{argmin}} \frac{\left(-B^{-1}\mathbf{b}\right)_t}{(B^{-1}N)_{tk}}$$

Note that the minimum is taken only over the positive components $\left(B^{-1}N\right)_{tk}$. If there are no positive components, the next corner is infinitely far away and the cost can be reduced forever to yield a minimum cost of $-\infty$.

5. With the new choice of basic variables, steps (2)-(4) are repeated till the reduced cost is completely non-negative. The variables corresponding to the unit columns in the final tableau are the basic variables at the optimum.

What we have not discussed so far is how to obtain the initial basic feasible point. If $\mathbf{x} = 0$ satisfies $A\mathbf{x} \geq \mathbf{b}$, we can have an initial basic feasible point with the basic variables comprising of $\mathbf{s}$ and $\mathbf{x}$ constituting the free variables. This is illustrated through the following example. Consider the problem

$$\begin{aligned}
\min_{x_1,x_2,x_3\in\Re} \quad & -15x_1 - 18x_2 - 20x_3 \\
\text{subject to} \quad & -\tfrac{1}{6}x_1 - \tfrac{1}{4}x_2 - \tfrac{1}{2}x_3 \geq -60 \\
& -40x_1 - 50x_2 - 60x_3 \geq -2880 \\
& -25x_1 - 30x_2 - 40x_3 \geq -2400 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0
\end{aligned}$$

The initial tableau is

$$\left[\begin{array}{cccccc|c}
\frac{1}{6} & \frac{1}{4} & \frac{1}{2} & 1 & 0 & 0 & 60 \\
40 & 50 & 60 & 0 & 1 & 0 & 2880 \\
25 & 30 & 40 & 0 & 0 & 1 & 2400 \\
-15 & -18 & -20 & 0 & 0 & 0 & 0
\end{array}\right]$$

The most negative component of the reduced cost vector is for $k = 3$. The pivot row number is $2 = \underset{t=1,2,\ldots,m \ \ \left(B^{-1}N\right)_{tk}>0}{\text{argmin}} \frac{\left(B^{-1}\mathbf{b}\right)_t}{\left(B^{-1}N\right)_{tk}}$. Thus, the leaving basic variable is s_2 (the basic variable corresponding to the second row) while the entering free variable is x_3. Performing Gauss elimination to obtain column $k = 3$ in the unit form, we get

$$\left[\begin{array}{cccccc|c}
-\frac{1}{6} & -\frac{1}{6} & 0 & 1 & -\frac{1}{120} & 0 & 36 \\
\frac{2}{3} & \frac{5}{6} & 1 & 0 & \frac{1}{60} & 0 & 48 \\
-\frac{5}{3} & -\frac{10}{3} & 0 & 0 & -\frac{2}{3} & 1 & 480 \\
-\frac{5}{3} & -\frac{4}{3} & 0 & 0 & \frac{1}{3} & 0 & 960
\end{array}\right]$$

This tableau corresponds to the solution $x_1 = 0, x_2 = 0, x_3 = 48, s_1 = 0, s_2 = 36, s_3 = 480$ and cost $\mathbf{c}^T\mathbf{x} = -960$ (negative of the number on the right hand bottom corner). Since the reduced cost vector has still some negative components, it is possible to find a basic feasible solution with lower cost. Using the most negative component of the reduced cost vector, we select the next pivot element to be $m_{21} = \frac{2}{3}$. Again performing Gaussian elimination, we obtain the tableau corresponding to the next iterate.

$$\left[\begin{array}{cccccc|c} 0 & \frac{1}{24} & \frac{1}{4} & 1 & -\frac{1}{240} & 0 & 48 \\ 1 & \frac{5}{4} & \frac{3}{2} & 0 & \frac{1}{40} & 0 & 72 \\ 0 & -\frac{5}{4} & \frac{5}{2} & 0 & -\frac{5}{8} & 1 & 600 \\ 0 & \frac{3}{4} & \frac{5}{2} & 0 & \frac{3}{8} & 0 & 1080 \end{array}\right]$$

Note that the optimal solution has been found, since the reduced cost vector is non-negative. The optimal solution is $x_1 = 72, x_2 = 0, x_3 = 0, s_1 = 48, s_2 = 0, s_3 = 600$ and cost $\mathbf{c}^T\mathbf{x} = -1080$

What if $\mathbf{x} = 0$ does not satisfy $A\mathbf{x} \geq \mathbf{b}$? The choice of $\mathbf{s}$ as the basic variables and $\mathbf{x}$ as the free variables will not be valid. As an example, consider the problem

$$\begin{array}{ll} \min\limits_{x_1,x_2,x_3\in\Re} & 30x_1 + 60x_2 + 70x_3 \\ \text{subject to} & x_1 + 3x_2 + 4x_3 \geq 14 \\ & 2x_1 + 2x_2 + 3x_3 \geq 16 \\ & x_1 + 3x_2 + 2x_3 \geq 12 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{array}$$

The initial tableau is

$$\left[\begin{array}{cccccc|c} -1 & -3 & -4 & 1 & 0 & 0 & -14 \\ -2 & -2 & -3 & 0 & 1 & 0 & -16 \\ -1 & -3 & -2 & 0 & 0 & 1 & -12 \\ 30 & 60 & 70 & 0 & 0 & 0 & 0 \end{array}\right]$$

With the choice of basic and free variables as above, we are not even in the feasible region to start off with. In general, if we have any negative number in the last column of the tableau, $\mathbf{x} = \mathbf{0}$ is not in the feasible region. Further, we have no negative numbers in the bottom row, which does not leave us with any choice of cost reducing free variable. But this is not of primary concern, since we first need to maneuver our way into the feasible region. We do this by moving from one basic point (that is, a point having not more than n zero components) to another till we land in the feasible region, which is indicated by all positive components in the extreme right hand column. This movement from one basic point to another is not driven by negative components in the cost vector, but rather by the negative components in the right hand column. The new rules for moving from one basic point to another are:

1. Pick[39] any negative number in the far right column (excluding the last row). Let this be in the q^{th} row for $q < m + 1$.

[39]Note that there is no priority here.

2. In the q^{th} row, move[40] to left to a column number k where there is another negative number. The variable y_k will be the next entering variable.
3. Choose pivot element m_{jk} which gives the smallest positive ratio of an element in the j^{th} row of the last column to the element m_{jk}. The leaving variable will be y_j.
4. Once the pivot element is chosen, proceed as usual to convert the pivot element to 1 and the other elements in the pivot column to 0.
5. Repeat steps (1)-(4) on the modified tableau until there is no negative element in the right-most column.

Applying this procedure to the tableau above, we pick $m_{2,1} = -2$ as our first pivot element and do row elimination to get the first column in unit form.

$$\left[\begin{array}{cccccc|c} 0 & -2 & -\frac{5}{2} & 1 & -\frac{1}{2} & 0 & -6 \\ 1 & 1 & \frac{3}{2} & 0 & -\frac{1}{2} & 0 & 8 \\ 0 & -2 & -\frac{1}{2} & 0 & -\frac{1}{2} & 1 & -4 \\ 0 & 30 & 25 & 0 & 15 & 0 & -240 \end{array}\right]$$

We pick $m_{35} = -\frac{1}{2}$ as our next pivot element and do similar row elimination operations to obtain

$$\left[\begin{array}{cccccc|c} 0 & 0 & -2 & 1 & 0 & -1 & -2 \\ 1 & 3 & 2 & 0 & 0 & -1 & 12 \\ 0 & 4 & 1 & 0 & 1 & -2 & 8 \\ 0 & -30 & 10 & 0 & 0 & 30 & -360 \end{array}\right]$$

We have still not obtained a feasible basic point. We choose $m_{16} = -1$ as the next pivot and do row eliminations to get the next tableau.

$$\left[\begin{array}{cccccc|c} 0 & 0 & 2 & -1 & 0 & 1 & 2 \\ 1 & 3 & 4 & -1 & 0 & 0 & 14 \\ 0 & 4 & 5 & -2 & 1 & 0 & 12 \\ 0 & -30 & -50 & 30 & 0 & 0 & -420 \end{array}\right]$$

This tableau has not negative numbers in the last column and gives a basic feasible point $x_1 = 14, x_2 = 0, x_3 = 0$. Once we obtain the basic feasible point, we rever to the standard simplex procedure discussed earlier. The most negative component of the reduced cost vector is -50 and this leads to the pivot element $m_{13} = 2$. Row elimination yields

$$\left[\begin{array}{cccccc|c} 0 & 0 & 1 & -\frac{1}{2} & 0 & \frac{1}{2} & 1 \\ 1 & 3 & 0 & 1 & 0 & -2 & 10 \\ 0 & 4 & 0 & \frac{1}{2} & 1 & -\frac{5}{2} & 7 \\ 0 & -30 & 0 & 5 & 0 & 25 & -370 \end{array}\right]$$

[40] Note that there is no priority here either.

Our next pivot element is $m_{32} = 4$. Row elimination yields

$$\left[\begin{array}{cccccc|c} 0 & 0 & 1 & -\frac{1}{2} & 0 & \frac{1}{2} & 1 \\ 1 & 0 & 0 & \frac{5}{8} & -\frac{3}{4} & -\frac{1}{8} & \frac{19}{4} \\ 0 & 1 & 0 & \frac{1}{8} & \frac{1}{4} & -\frac{5}{8} & \frac{7}{4} \\ 0 & 0 & 0 & \frac{35}{4} & \frac{15}{2} & \frac{25}{4} & -\frac{635}{2} \end{array}\right]$$

We are done! The reduced cost vector has no more negative components. The optimal basic feasible point is $x_1 = 4.75, x_2 = 1.75, x_3 = 1$ and the optimal cost is 317.5.

Revised Simplex Method

The simplex method illustrated above serves two purposes:

1. Doing all the eliminations completely makes the idea clear.

2. It easier to follow the process when working out the solution by hand.

For computational purposes however, it is uncommon now to use the method as described earlier. This is because, once $\mathbf{r}$ is computed, none of the columns above $\mathbf{r}$, (except for that corresponding to the leaving variable) are used. Therefore, computing them is a useless effort. Doing the eliminations completely at each step cannot be justified practically. Instead, the more efficient version of the simplex method, as outlined below, is used by software packages. It is called the revised simplex method and is essentially the simplex method itself, boiled down.

```
Compute the reduced costs $\mathbf{r} = \mathbf{d}_N - (\mathbf{d}_B)B^{-1}N$.
while $\mathbf{r} \ngeq \mathbf{0}$ do
    1. Let $r_k$ be the most negative component of $\mathbf{r}$.
    2. Compute $\mathbf{v} = B^{-1}\mathbf{n}_i$, where $\mathbf{n}_i$ is the $i^{th}$ column of $N$.
    3. Let $j = \underset{t=1,2,\ldots,m \quad (B^{-1}\mathbf{n}_i)_t > 0}{\text{argmin}} \frac{(-B^{-1}\mathbf{b})_t}{(B^{-1}\mathbf{n}_i)_t}$.
    4. Update $B$ (or $B^{-1}$) and $\mathbf{x}_B = B^{-1}\mathbf{b}$ to reflect the $j^{th}$ leaving column
    and the $k^{th}$ entering variable.
    Compute the new reduced costs $\mathbf{r} = \mathbf{d}_N - (\mathbf{d}_B)B^{-1}N$.
end while
```

Figure 3.58: The revised simplex method.

3.7.2 Interior point barrier method

Researchers have dreamt up pathological examples for the simplex method, for which the simplex method takes an exponential amount of time. In practice, however, the simplex method is one of the most efficient methods for a majority of the LPs. Application of interior point methods to LP have led to a new competitor to the simplex method in the form of interior point methods for linear programming. In contrast to the simplex algorithm, which finds the optimal solution by progressing along points on the boundary of a polyhedral set, interior point methods traverse to the optimal through the interior of the feasible region (polyhedral set in the case of LPs). The first in this league was the iterative Karmarkar's method [**?**], developed by Narendra Karmarkar in 1984. Karmarkar also proved that the algorithm was polynomial time. This line of research was inspired by the *ellipsoid method* for linear programming, outlined by Leonid Khachiyan in 1979; the ellipsoid algorithm itself was introduced by Naum Z. Shor, *et. al.* in 1972 and used by Leonid Khachiyan [**?**] to prove the polynomial-time solvability of linear programs r linear programming , which was the first such algorithm known to have a polynomial running time.

This competitor to the simplex method takes a Newton's method-like approach through the interior of the feasible region. Newton steps are taken till the 'barrier' is encountered. It stirred up the world of optimization and inspired the whole class of barrier methods. Following this, a lot of interest was generated in the application of the erstwhile interior point methods for general non-linear constrainted optimization problems. The Karmarkar's algorithm is now replaced by an improved logarithmic barrier method that makes use of the primal as well as the dual for solving an LP. Shanno and Bagchi [**?**] showed that Karmarkars method is just a special case of the logarithmic barrier function method. We will restrict our discussion to the primal-dual barrier method [**?**].

The dual (3.113) for the linear program (3.111) can be derived in manner similar to the dual on page 233.

$$\begin{aligned}
\max_{\lambda \in \Re^m} \quad & \lambda^T \mathbf{b} \\
\text{subject to} \quad & A^T \lambda \leq \mathbf{c} \\
& \lambda \geq \mathbf{0}
\end{aligned}$$

The weak duality theorem (theorem 66) states that the objective function value of the dual at any feasible solution is always less than or equal to the objective function value of the primal at any feasible solution. That is, for any primal feasible $\mathbf{x}$ and any dual feasible λ,

$$\mathbf{c}^T\mathbf{x} - \mathbf{b}^T\lambda \geq 0$$

For this specific case, the weak duality is easy to see: $\mathbf{b}^T\lambda \leq \mathbf{x}^T A^T \lambda \leq \mathbf{x}^T\mathbf{c}$.

Further, it can be proved using the *Farkas' lemma* that if the primal has an optimal solution $\mathbf{x}^*$ (which is assumed to be bounded), then the dual also has an optimal solution[41] λ^*, such that $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\lambda^*$.

The following steps will set the platform for the interior point method.

[41]For an LP and its dual D, there are only four possibilities:

1. (LP) is bounded and feasible and (D) is bounded and feasible.

1. As on page 3.7.1, we will rewrite the constraints $A\mathbf{x} \geq \mathbf{b}$ in the above LP as equations, by introducing a new non-negative "slack" variable s_j for the j^{th} constraint (for all j's) and subtracting it from the left-hand side of each inequality. This gives us the constraint $M\mathbf{y} = -\mathbf{b}$, where $M = [-A \;\; I]$ and $\mathbf{y} = [\mathbf{x} \;\; \mathbf{s}]^T$. As before, the original cost vector $\mathbf{c}$ is extended to a vector $\mathbf{d}$ by appending m more zero components. This gives us the following problem, equivalent to the original LP (**??**).

$$\begin{aligned} \min_{\mathbf{y}\in\Re^{n+m}} \quad & \mathbf{y}^T\mathbf{d} \\ \text{subject to} \quad & M\mathbf{y} = -\mathbf{b} \\ & \mathbf{y} \geq \mathbf{0} \end{aligned} \tag{3.113}$$

Its dual problem is given by

$$\begin{aligned} \max_{\lambda\in\Re^{m}} \quad & -\lambda^T\mathbf{b} \\ \text{subject to} \quad & M^T\lambda \leq \mathbf{d} \end{aligned} \tag{3.114}$$

2. Next, we set up the barrier method formulation of the dual of the linear program. Letting $\mu > 0$ be a given fixed parameter, which is decreased during the course of the algorithm. We also insert slack variables $\xi = [\xi_1, \xi_2, \ldots, \xi_n]^T \geq \mathbf{0}$. The barrier method formulation of the dual is then given by:

$$\begin{aligned} \max_{\lambda\in\Re^{m}} \quad & -\lambda^T\mathbf{b} + \mu\textstyle\sum_{i=1}^{n} \ln(\xi_i) \\ \text{subject to} \quad & M^T\lambda + \xi = \mathbf{d} \end{aligned} \tag{3.115}$$

The conditions $\xi_i \geq 0$ are no longer needed since $\ln(\xi_i) \to \infty$ as $\xi_i \to 0$, if $\xi_i > 0$. This latter property means that $\ln(\xi_i)$ serves as a barrier, discouraging ξ_i from going to 0.

3. To write the first-order necessary conditions for a minimum, we set the partial derivatives of the Lagrangian

$$L(\mathbf{y}, \lambda, \xi) = -\lambda^T\mathbf{b} + \mu\sum_{i=1}^{n} \ln(\xi_i) - \mathbf{y}^T\left(M^T\lambda + \xi - \mathbf{d}\right)$$

2. (LP) is infeasible and (D) is unbounded and feasible.
3. (LP) is unbounded and feasible and (D) is infeasible.
4. (LP) is infeasible and (D) is infeasible.

This can be proved using the *Farkas' Lemma*.

with respect to $\mathbf{y}$, λ, ξ to zero. This results in the set of following three equations:

$$\begin{aligned} &M^T\lambda + \xi = \mathbf{d} \\ &M\mathbf{y} = -\mathbf{b} \\ &\operatorname{diag}(\xi)\operatorname{diag}(\mathbf{y})\mathbf{1} = \mu\mathbf{1} \end{aligned} \tag{3.116}$$

which include the dual and primal feasibility conditions excluding $\mathbf{y} \geq \mathbf{0}$ and $\xi \geq \mathbf{0}$.

4. We will assume that our current point $\mathbf{y}^{(k)}$ is primal feasible and the current point $(\lambda^{(k)}, \xi^{(k)})$ is dual feasible. We determine a new search direction $\left(\Delta\mathbf{y}^{(k)}, \Delta\lambda^{(k)}, \Delta\xi^{(k)}\right)$ so that the new point $\left(\mathbf{y}^{(k)} + \Delta\mathbf{y}^{(k)}, \lambda^{(k)} + \Delta\lambda^{(k)}, \xi^{(k)} + \Delta\xi)^{(k)}\right)$ satisfies (3.116). This gives us the so-called Newton equations:

$$\begin{aligned} &M^T\Delta\lambda + \Delta\xi = \mathbf{0} \\ &M\Delta\mathbf{y} = \mathbf{0} \\ &(y_i + \Delta y_i)(\xi_i + \Delta\xi_i) = \mu \quad i = 1, 2, \ldots, n \end{aligned} \tag{3.117}$$

Ignoring the second order term $\Delta y_i \Delta\xi_i$ in the third equation, and solving the system of equations in (3.117), we get the following update rules:

$$\begin{aligned} \Delta\lambda^{(k)} &= -\left(M\operatorname{diag}(\mathbf{y}^{(k)})\operatorname{diag}(\xi^{(k)})^{-1}M^T\right)^{-1} M\operatorname{diag}(\xi^{(k)})^{-1}\left(\mu\mathbf{1} - \operatorname{diag}(\mathbf{y}^{(k)})\operatorname{diag}(\xi^{(k)})\mathbf{1}\right) \\ \Delta\xi^{(k)} &= -M^T\Delta\lambda^{(k)} \\ \Delta\mathbf{y}^{(k)} &= \operatorname{diag}(\xi^{(k)})^{-1}\left(\mu\mathbf{1} - \operatorname{diag}(\mathbf{y}^{(k)})\operatorname{diag}(\xi^{(k)})\mathbf{1}\right) - \operatorname{diag}(\mathbf{y}^{(k)})\operatorname{diag}(\xi^{(k)})^{-1}\Delta\xi^{(k)} \end{aligned} \tag{3.118}$$

An affine variant of the algorithm can be developed by setting $\mu = 0$ in the equations (3.118).

5. $\Delta\mathbf{y}^{(k)}$ and $(\Delta\lambda^{(k)}, \Delta\xi^{(k)})$ correspond to partially constrained Newton steps, which might not honour the constraints $\mathbf{y}^{(k)}+\Delta\mathbf{y}^{(k)} \geq \mathbf{0}$ and $\xi^{(k)}+\Delta\xi^{(k)} \geq \mathbf{0}$. Since we have a separate search direction $\Delta\mathbf{y}^{(k)}$ in the primal space and a separate search direction $(\Delta\lambda^{(k)}, \Delta\xi^{(k)})$ in the dual space, we could compute the maximum step length $t^{(k)}_{(max,P)}$ that maintains the pending primal inequality $\mathbf{y}^{(k)}+t^{(k)}_{(max,P)}\Delta\mathbf{y}^{(k)} \geq \mathbf{0}$ and the maximum step length $t^{(k)}_{(max,D)}$ that maintains the pending dual inequality $\xi^{(k)} + t^{(k)}_{(max,D)}\Delta\xi^{(k)} \geq \mathbf{0}$.

6. Now we have a feasible primal solution $\mathbf{y}^{(k+1)}$ and feasible dual solution $(\lambda^{(k+1)}, \xi^{(k+1)})$ given by

$$\begin{aligned}
\mathbf{y}^{(k+1)} &= \mathbf{y}^{(k)} + t^{(k)}_{(max,P)}\Delta\mathbf{y}^{(k)} \\
\lambda^{(k+1)} &= \lambda^{(k)} + \Delta\lambda^{(k)} \\
\xi^{(k+1)} &= \xi^{(k)} + t^{(k)}_{(max,D)}\Delta\xi^{(k)}
\end{aligned} \tag{3.119}$$

7. For user specified small thresholds of $\epsilon_1 > 0$ and $\epsilon_2 > 0$, if the duality gap $\mathbf{d}^T\mathbf{y}^{(k+1)} + \mathbf{b}^T\lambda^{(k+1)}$ is not sufficiently close to 0, *i.e.*,

   $$\mathbf{d}^T\mathbf{y}^{(k+1)} + \mathbf{b}^T\lambda^{(k+1)} > \epsilon_1$$

   for a μ not yet sufficiently close to 0, *i.e.*,

   $$\mu > \epsilon_2$$

   we decrease μ by a user specified factor $\rho < 1$ (such as $\rho = 0.1$).

   $$\mu = \mu \times \rho$$

8. Set $k = k + 1$. If μ was not modified in step (7), the duality gap is sufficiently small and the termination condition has been reached. So EXIT. Else, the last condition in (3.116) no longer holds with the modified value of μ. So steps (4)-(7) are re-executed.

In practice, the interior point method for LP gets down the duality gap to within 10^{-8} in just 20-80 steps (which is still slower than the simplex method for many problems), independent of the size of the problem specified through values of m and n.

3.8 Least Squares

Least squares was motivated in Section 2.9.2, based on the idea of projection. Least squares problems appear very frequently in practice. The objective for minimization in the case of least squares is the square of the eucledian norm of $A\mathbf{x} - \mathbf{b}$, where A is a $m \times n$ matrix, $\mathbf{x}$ is a vector of n variables and $\mathbf{b}$ is a vector of m knowns.

$$\min_{\mathbf{x}\in\Re^n} \quad ||A\mathbf{x} - \mathbf{b}||_2^2 \tag{3.120}$$

Very often one has a system of linear constraints on problem (3.120).

$$\begin{aligned}
&\min_{\mathbf{x}\in\Re^n} && ||A\mathbf{x} - \mathbf{b}||_2^2 \\
&\text{subject to} && C^T\mathbf{x} = \mathbf{0}
\end{aligned} \tag{3.121}$$

This problem is called the *least squares problem with linear constraints.*

In practice, incorporating the constraints $C^T\mathbf{x} = \mathbf{0}$ properlymakes quite a difference. In lots of regularization problems, the least squares problem often comes with quadratic constraints in the following form.

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & ||A\mathbf{x} - \mathbf{b}||_2^2 \\ \text{subject to} & ||\mathbf{x}||_2^2 = \alpha^2 \end{array} \tag{3.122}$$

This problem is termed as the *least squares problem with quadratic constraints.*

The classical statistical model assumes that all the error occurs in the vector $\mathbf{b}$. But sometimes, the data matrix A is itself not very well known, owing to errors in the variables. This is the model we have in the simplest version of the *total least squares problem*, which is stated as follows.

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n, E\in\Re^{m\times n}, \mathbf{r}\in\Re^m} & ||E||_F^2 + ||\mathbf{r}||_2^2 \\ \text{subject to} & (A+E)\mathbf{x} = \mathbf{b} + \mathbf{r} \end{array} \tag{3.123}$$

While there is always a solution to the least squares problem (3.120), there is not always a solution to the total least squares problem (3.133). Finally, you can have a combination of linear and quadratic constraints in a least squares problem to yield a *least squares problem with linear and quadratic constraints.*

We will briefly discuss the problem of solving linear least squares problems and total least squares problems with linear or a quadratic constraint (due to regularization) The importance of lagrange multipliers will be introduced in the process. We will discuss stable numerical methods when the data matrix A is singular or near singular. We will also present iterative methods for large and sparse data matrices. There are many applications of least squares problems, which include statistical methods, image processing, data interpolation and surface fitting and finally geometrical problems.

3.8.1 Linear Least Squares

As a user of least squares in practice, one of the most important things to be known is that when A is of full column rank, it has an analytical solution given by $\mathbf{x}^*$ (which was derived in Section 2.9.2 and gives a dual interpretation).

$$\text{Analytical solution:} \quad \mathbf{x}^* = (A^TA)^{-1}A^T\mathbf{b} \tag{3.124}$$

This analytic solution can also be obtained by observing that

1. $||\mathbf{y}||_2^2$ is a convex function for $\mathbf{y} \in \Re^m$.

2. Square of the convex eucledian norm function, applied to an affine transform is also convex. Thus $||A\mathbf{x} - \mathbf{b}||_2^2$ is convex.

3. Every critical point of a convex function defined on an open domain corresponds to its local minimum. The critical point $\mathbf{x}^*$ of $||A\mathbf{x} - \mathbf{b}||_2^2$ should satisfy
$$\nabla(A\mathbf{x} - \mathbf{b})^T(A\mathbf{x} - \mathbf{b}) = 2A^TA\mathbf{x}^* - 2A^T\mathbf{b} = \mathbf{0}$$
Thus,
$$\mathbf{x}^* = (A^TA)^{-1}A^T\mathbf{b}$$
corresponds to a point of local minumum of (3.120) if A^TA is invertible.

This is the classical way statisticians solve least squares problem. It can be solved very efficiently, and there exist many softwares that implement this solution. The computation time is linear in the number of rows of A and quadratic in the number of columns. For extremely large A, it can become important to look at the structure of A to solve it efficiently, but for most problems, it is efficient. In practice least-squares is very easy to recognize as an objective function. There are a few standard tricks to increase the flexibility. For example, constraints can be handled to a certain extent by adding weights. When the matrix A is not full column rank, the solution to (3.120) may not be unique.

We should note that while we get a closed form solution to the problem of minimizing the square of the eucledian norm, it is not so for most other norms such as the infinity norm. However, there exist iterative methods for solving least squares with infinity norm that yield a solution in as much time as is taken in computing the solution using the analytical formula in 3.124. Therefore, having a closed form solution is not always computationally helpful. In general, the method of solution to a least squares problem depends on the sparsity as well as the size of A and the degree of accuracy desired.

In practice, however, it is not recommended to solve least squares problem using the classical equation in 3.124 since the method is numerically unstable. Numerical linear algebra instead recommends the QR decomposition to accurately solve the least squares problem. This method is slower, but more numerically stable than the classical method. In theorem **??**, we state a theory that compares the analytical solution (3.124) and the QR approach to the least squares problem.

Let A be an $m \times n$ matrix of either full row or full column rank. For the case of $n > m$, we saw on page **??** (summarised in Figure 2.3) that the system $A\mathbf{x} = \mathbf{b}$ will have at least one solution which means that minimum value of the objective function will be 0, corresponding to the solution. We are interested in the case $m \geq n$, for which there will either be no solution or a single solution to $A\mathbf{x} = \mathbf{b}$ and we are interested in one that minimizes $||A\mathbf{x} - \mathbf{b}||_2^2$.

1. We first decompose A into the product of an orthonormal $m \times m$ matrix Q with an upper traingular $m \times n$ matrix R, using the gram-schmidt or-

thonormalization process[42] discussed in Section 2.9.4. The decomposition can also be performed using the Householder[43] transformation or Givens rotation. Householder transformation has the added advantage that new rows or columns can be introduced without requiring a complete redo of the decomposition process. The last $m-n$ rows of R will be zero rows. Since $Q^{-1}=Q^T$, the QR decomposition yields the system

$$Q^T A = \begin{bmatrix} R_1 \\ \mathbf{0} \end{bmatrix}$$

2. Applying the same orthogonal matrix to $\mathbf{b}$, we get

$$Q^T \mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

where $\mathbf{d} \in \Re^{m-n}$.

3. The solution to the least squares problem is found by solving $R_1\mathbf{x}=\mathbf{c}$. The solution to this can be found by simple back-substitution.

The next theorem examines how the least squares solution and its residual $||A\mathbf{x}-\mathbf{b}||$ are affected by changes in A and $\mathbf{b}$. Before stating the theorem, we will introduce the concept of the condition number.

Condition Number

The *condition number* associated with a problem is a measure of how numerically well-posed the problem is. A problem with a low condition number is said to be well-conditioned, while a problem with a high condition number is said to be ill-conditioned. For a linear system $A\mathbf{x}=\mathbf{b}$, the condition number is defined as maximum ratio of the relative error in $\mathbf{x}$ (measured using any particular norm) divided by the relative error in $\mathbf{b}$. It can be proved (using the Cauchy Shwarz inequality) that the condition number equals $||A^{-1}A||$ and is independent of $\mathbf{b}$. It is denoted by $\kappa(A)$ and is also called the condition number of the matrix A.

$$\kappa(A) = ||A^{-1}A||$$

If $||.||_2$ is the L_2 norm, then

$$\kappa(A) = \frac{\sigma_{max}(A)}{\sigma_{min}(A)} = ||A||_2||(A^TA)^{-1}A^T||_2$$

[42]The classical Gram-Schmidt method is often numerically unstable. Golub [?] suggests a modified Gram-Schmidt method that is numerically stable.

[43]Householder was a numerical analyst. However, the first mention of the Householder transformation dates back to the 1930s in a book by Aikins, a statistician and a numerical analyst.

where $\sigma_{max}(A)$ and $\sigma_{min}(A)$ are maximal and minimal singular values of A respectively. For a real square matrix A, the square roots of the eigenvalues of A^TA, are called singular values. Further,

$$\kappa(A)^2 = ||A||_2^2||(A^TA)^{-1}||_2^2$$

Theorem 71 *By $||.||$, we will refer to the L_2 norm. Let*

$$\mathbf{x}^* = \text{argmin } \; ||A\mathbf{x} - \mathbf{b}||$$

$$\widehat{\mathbf{x}} = \text{argmin } \; ||(A + \delta A)\mathbf{x} - (\mathbf{b} + \delta\mathbf{b}||$$

where A and δA are in $\Re^{m\times n}$ with $m \geq n$. Let $\mathbf{b}$ and $\delta\mathbf{b}$ be in $\Re^m$ with $\mathbf{b} \neq \mathbf{0}$. Let us set

$$\mathbf{r}^* = \mathbf{b} - A\mathbf{x}^*$$

$$\widehat{\mathbf{r}} = \mathbf{b} - A\widehat{\mathbf{x}}$$

and

$$\rho^* = ||A\mathbf{x}^* - \mathbf{b}||$$

If

$$\epsilon = max\left\{\frac{||\delta A||}{||A||}, \frac{\delta\mathbf{b}}{\mathbf{b}}\right\} < \frac{\sigma_n(A)}{\sigma_1(A)}$$

and

$$\sin\theta = \frac{\rho^*}{||\mathbf{b}||} \neq 1$$

then,

$$\frac{||\widehat{\mathbf{x}} - \mathbf{x}^*||}{||\mathbf{x}^*||} \leq \epsilon\left\{\frac{2\kappa(A)}{\cos\theta} + \tan\theta\kappa(A)^2\right\} + O(\epsilon^2)$$

In this inequality most critical term for our discussion is $\kappa(A)^2$ and this is the term that can kill the analytical solution to least squares. Now matter how accurate an algorithm you use, you still have $\kappa(A)^2$, provided $\tan\theta$ is non-zero. Now $\tan\theta$ does not appear if you are solving a linear system, but if you solve a least squares problem this term appears, bringing along $\kappa(A)^2$. Thus, solving least squares problem is inherently more difficult and sensitive than linear equations. The perturbation theory for the residual vector depends just on the condition number $\kappa(A)$ (and not its square):

$$\frac{||\widehat{\mathbf{r}} - \mathbf{r}^*||}{||\mathbf{b}||} \leq \epsilon\left\{1 + 2\kappa(A)\right\} min\left\{1, m - n\right\} + O(\epsilon^2) + O(\epsilon^2)$$

However, having a small residual does not necessarily imply that you will have a good approximate solution.

The theorem implies that the sensitivity of the analytical solution $\mathbf{x}^*$ for non-zero residual problems is measured by the square of the condition number. Whereas, sensitivity of the residual depends just linearly on $\kappa(A)$. We note that the QR method actually solves a nearby least squares problem.

Linear Least Squares for Singular Systems

To solve the linear least squares problem (3.120) for a matrix A that is of rank $r < min\{m,n\}$, we can compute the pseudo-inverse (*c.f.* page 154) A^+ and obtain the least squares solution[44] as

$$\widehat{\mathbf{x}} = A^+\mathbf{b}$$

A^+ can be computed by first computing a singular orthogonal factorization

$$A = Q\begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix}Z^T$$

where $Q^TQ = I_{m\times m}$ and $Z^TZ = I_{n\times n}$ and R is an $r \times r$ upper traingular matrix. A^+ can be computed in a straightforward manner as

$$A^+ = Z\begin{bmatrix} R^{-1} & 0 \\ 0 & 0 \end{bmatrix}Q^T$$

The above least squares solution can be justified as follows. Let

$$Q^T\mathbf{b} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

and

$$Z^T\mathbf{x} = \begin{bmatrix} \mathbf{w} \\ \mathbf{y} \end{bmatrix}$$

Then

$$||A\mathbf{x} - \mathbf{b}||^2 = ||Q^TAZZ^T\mathbf{x} - Q^T\mathbf{b}||^2 = ||R\mathbf{w} - \mathbf{c}||^2 + ||\mathbf{d}||^2$$

The least squares solution is therefore given by

$$\widehat{\mathbf{x}} = Z\begin{bmatrix} R^{-1}\mathbf{c} \\ 0 \end{bmatrix}$$

One particular decomposition that can be used is the singular value decomposition (*c.f.* Section 2.13) of A, with $Q^T \equiv U^T$ and $Z \equiv V$ and $U^TAV = \Sigma$. The pseudo-inverse A^+ has the following expression.

$$A^+ = V\Sigma^{-1}U^T$$

It can be shown that this A^+ is the unique minimal Frobenius norm solution to the following problem.

$$A^+ = \underset{X\in\Re^{n\times m}}{\text{argmin}}\ ||AX - I_{m\times m}||$$

[44]Note that this solution not only minimizes $||A\mathbf{x} - \mathbf{b}||$ but also minimizes $||\mathbf{x}||$. This may or may not be desirable.

This also shows that singular value decomposition can be looked upon as an optimization problem.

A greater problem is with systems that are nearly singular. Numerically and computationally it seldom happens that the rank of matrix is exactly r. A classical example is the following $n \times n$ matrix K, which has a determinant of 1.

$$K = \begin{bmatrix} 1 & -1 & \ldots & -1 & -1 & \ldots & -1 \\ 0 & 1 & \ldots & -1 & -1 & \ldots & -1 \\ . & . & \ldots & . & . & \ldots & . \\ . & . & \ldots & . & . & \ldots & . \\ 0 & 0 & \ldots & 1 & -1 & \ldots & -1 \\ 0 & 0 & \ldots & 0 & 1 & \ldots & -1 \\ . & . & \ldots & . & . & \ldots & . \\ . & . & \ldots & . & . & \ldots & . \\ 0 & 0 & \ldots & 0 & 0 & \ldots & 1 \end{bmatrix}$$

The eigenvalues of this matrix are also equal to 1, while its rank is n. However, a very small perturbation to this matrix can reduce its rank to $n-1$; the rank of $K - 2^{-(n-1)} I_{n\times n}$ is $n-1$! Such catastrophic problems occur very often when you do large computations. The solution using SVD is applicable for nearly singular systems as well.

3.8.2 Least Squares with Linear Constraints

We first reproduce the least squares problem with linear constraints that was stated earlier in (3.121).

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & ||A\mathbf{x}-\mathbf{b}||^2 \\ \text{subject to} & C^T\mathbf{x} = \mathbf{0} \end{array}$$

Let $C \in \Re^{n\times p}$, $A \in \Re^{m\times n}$ and $\mathbf{b} \in \Re^m$. We note that $||A\mathbf{x}-\mathbf{b}||^2$ is a convex function (since L_2 norm is convex and this function is the L_2 norm applied to an affine transform). We can thus solve this constrained problem by invoking the necessary and sufficient KKT conditions discussed in Section 3.4.4. The conditions can be worked out to yield

$$\begin{bmatrix} A^TA & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \lambda \end{bmatrix} = \begin{bmatrix} A^T\mathbf{b} \\ \mathbf{0} \end{bmatrix}$$

We need to now solve not only for the unknowns $\mathbf{x}$, but also for the lagrange multipliers; we have increased the dimensionality of the problem to $n+p$. If $\widehat{\mathbf{x}} =$

$(A^TA)^{-1}A^T\mathbf{b}$ denotes the solution of the unconstrained least squares problem, then, using the first system of equality above, $\mathbf{x}$ can be expressed as

$$\mathbf{x} = \widehat{\mathbf{x}} - (A^TA)^{-1}C\lambda \tag{3.125}$$

In conjunction with the second system, this leads to

$$C^T(A^TA)^{-1}C\lambda = C^T\widehat{\mathbf{x}} \tag{3.126}$$

The unconstrained least squares solution can be obtained using methods in Section 3.8.1. Next, the value of λ can be obtained by solving (3.126). If A is singular or nearly singular, we can use the singular value decomposition (or a similar decomposition) of A to determine $\widehat{\mathbf{x}}$.

$$C^TR^{-1}(R^T)^{-1}C\lambda = C^T\widehat{\mathbf{x}}$$

The QR factorization of $(R^T)^{-1}C$ can be efficiently used to determine λ. Finally, the value of λ can be substituted in (3.125) to solve for $\mathbf{x}$. This technique yields both the solutions, provided that both exist.

Another trick that is often employed when A^TA is singular or nearly singular is to decrease its condition number by augmenting it in (3.125) with the 'harmless' CWC^T and solve

$$\mathbf{x} = \widehat{\mathbf{x}} - (A^TA + CWC^T)^{-1}C\lambda$$

The addition of CWC^T is considered harmless, since $C^T\mathbf{x} = 0$ is to be imposed anyways. Matrix W can be chosen to be an identical or nearly identical matrix that chooses a few columns of C, just to make $A^TA + CWC^T$ non-singular.

If we use the following notation:

$$\mathcal{A}(W) = \begin{bmatrix} A^TA + CWC^T & C \\ C^T & 0 \end{bmatrix}$$

and

$$\mathcal{A} = \mathcal{A}(0) = \begin{bmatrix} A^TA & C \\ C^T & 0 \end{bmatrix}$$

and if $\mathcal{A}$ and $\mathcal{A}(W)$ are invertible for $W \neq 0$, it can be proved that

$$\mathcal{A}^{-1}(W) = \mathcal{A}^{-1} - \begin{bmatrix} 0 & 0 \\ 0 & W \end{bmatrix}$$

Consequently

$$\kappa(\mathcal{A}(W)) \leq \kappa(\mathcal{A}) + ||W||^2||C||^2 + \alpha||W||$$

for some $\alpha > 0$. That is, the condition number of $\mathcal{A}(W)$ is bounded by the condition number of $\mathcal{A}$ and some positive terms.

Another useful technique is to find an approximation to (3.121) by solving the following weighted unconstrained minimization problem.

$$\min_{\mathbf{x}\in\Re^n} \quad ||A\mathbf{x}-\mathbf{b}||^2+\mu^2||C^T\mathbf{x}||^2$$

For large values of μ, the solution $\widehat{\mathbf{x}}(\mu)$ of the unconstrained problem is a good appproximation to the solution $\widehat{\mathbf{x}}$ of the constrained problem (3.121). We can use the generalized singular value decompositions of matrices A and C^T, that allows us to simultaneously diagonalize A and C^T.

$$U^TAX=\mathbf{d}iag(\alpha_1,\ldots,\alpha_m)$$
$$V^TC^TX=\mathbf{d}iag(\gamma_1,\ldots,\gamma_m)$$

where U and V are orthogonal matrices and X is some general matrix. The solution to the constrained problem can be expressed as

$$\widehat{\mathbf{x}}=\sum_{i=1}^{p}\frac{\mathbf{u}_i^T\mathbf{b}}{\alpha_i}\mathbf{x}_i$$

The analytical solution $\widehat{\mathbf{x}}(\mu)$ is then given as

$$\widehat{\mathbf{x}}(\mu)=\sum_{i=1}^{p}\frac{\alpha_i\mathbf{u}_i^T\mathbf{b}}{\alpha_i^2+\mu^2\gamma_i^2}\mathbf{x}_i+\widehat{\mathbf{x}}$$

It can be easily seen that as $\mu^2\to\infty$, $\widehat{\mathbf{x}}(\mu)\to\widehat{\mathbf{x}}$.

Generally, if possible, it is better to eliminate the constraints, since this makes the problem better conditioned. We will discuss one final approach to solving the linearly constrained least squares problem (3.121), which reduces the dimensionality of the problem by eliminating the constraints. It is hinged on computing the QR factorization of C.

$$Q^TC=\begin{array}{c c}\left(\begin{array}{c} R \\ 0 \end{array}\right) & \begin{array}{l} p \\ n-p \end{array}\end{array} \tag{3.127}$$

This yields

$$AQ^T=\left(\begin{array}{cc} A_1 & A_2 \end{array}\right) \tag{3.128}$$

and

$$Q^T\mathbf{x}=\begin{array}{c c}\left(\begin{array}{c} \mathbf{y} \\ \mathbf{z} \end{array}\right) & \begin{array}{l} p \\ n-p \end{array}\end{array} \tag{3.129}$$

The constrained problem then becomes

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & ||\mathbf{b} - A_1\mathbf{y} - A_2\mathbf{z}||^2 \\ \text{subject to} & R^T\mathbf{y} = \mathbf{0} \end{array}$$

Since R is invertible, we must have $\mathbf{y} = \mathbf{0}$. Thus, the solution $\widehat{\mathbf{x}}$ to the constrained least squares problem can be determined as

$$\widehat{\mathbf{x}} = Q^T \begin{pmatrix} \mathbf{0} \\ \widehat{\mathbf{z}} \end{pmatrix} \tag{3.130}$$

where

$$\widehat{\mathbf{z}} = \underset{\mathbf{z}}{\text{argmax}}\ ||\mathbf{b} - A_2\mathbf{z}||^2$$

It can be proved that the matrix A_2 is atleast as well-conditioned as the matrix A. Often, the original problem is singular and imposing the constraints makes it non-singular (and is reflected in a non-singular matrix A_2.

3.8.3 Least Squares with Quadratic Constraints

The quadratically constrained least squares problem is often encountered in regularization problems and can be stated as follows.

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n} & ||A\mathbf{x} - \mathbf{b}||_2^2 \\ \text{subject to} & ||\mathbf{x}||_2^2 = \alpha^2 \end{array}$$

Since the objective function as well as the constraint function are convex, the KKT conditions (*c.f.* Section 3.4.4) are necessary and sufficient conditions for the optimality of the problem at the primal-dual variable pair given by $(\widehat{\mathbf{x}}, \widehat{\mu})$. The KKT conditions lead to the following equations

$$(A^TA + \mu I)\mathbf{x} = A^T\mathbf{b} \tag{3.131}$$

$$\mathbf{x}^T\mathbf{x} = \alpha^2 \tag{3.132}$$

The expression in (3.131) is the solution to what the statisticians sometimes refer to as the ridge regression problem. The solution to the problem under consideration has the additional constraint though, that the norm of the solution vector $\widehat{\mathbf{x}}$ should equal $|\alpha|$. The two equations above yield the so-called *secular equation* stated below.

$$\mathbf{b}^TA(A^TA + \mu I)^{-2}A^T\mathbf{b} - \alpha^2 = 0$$

Further, the matrix A can be diagonalized using its singular value decomposition $A = U\Sigma V^T$ to obtain the following equation which is to be solved.

$$\sum_{i=1}^{n} \beta_i^2 \frac{\sigma_i^2}{(\sigma_i^2 + \mu)^2} - \alpha^2 = 0$$

3.8.4 Total Least Squares

The total least squares problem is stated as

$$\begin{array}{ll} \min\limits_{\mathbf{x}\in\Re^n, E\in\Re^{m\times n}, \mathbf{r}\in\Re^m} & ||E||_F^2 + ||\mathbf{r}||_2^2 \\ \text{subject to} & (A+E)\mathbf{x} = \mathbf{b} + \mathbf{r} \end{array}$$

References

[CGH97] E. Castillo, J. M. Gutierrez, and A. S. Hadi. *Expert systems and probabilistic network models.* Springer-Verlag, 1997.

[Jen01] F. V. Jensen. *Bayesian Networks and Decision Graphs.* Springer, 2001.

[Jor98] M. I. Jordan. *Learning in Graphical Models.* MIT Press, 1998.

[Lau96] S. L. Lauritzen. *Graphical Models.* Clarendon Press, Oxford, 1996.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[SS90] Glenn Shafer and Prakash P. Shenoy. Probability propagation. *Ann. Math. Artif. Intell.*, 2:327–351, 1990.