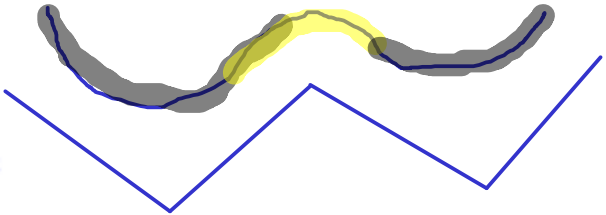
Lecture 1 : Introduction to Convex Optimization CS709

Instructor: Prof. Ganesh Ramakrishnan

Introduction: Mathematical optimization

- **Motivating Example**
- **Applications**
- **Least-squares(LS) and linear programming(LP) - Very common place**



- **Course goals and topics**
- **Nonlinear optimization**
- **Brief history of convex optimization**

Mathematical optimization

(Mathematical) Optimization problem:-

$$\underset{x}{\text{minimize}} \quad f_0(x)$$

$$\text{subject to} \quad f_i(x) \leq b_i, \; i = 1, \ldots, m.$$

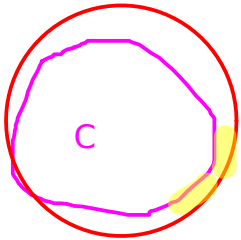
$x = (x_1, ..., x_n)$: optimization variables

$f_i : \Re^n \rightarrow \Re, \; i = 1, ..., m$: constraint functions

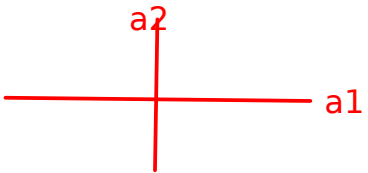
optimal solution x^* has smallest value of f_0 among all vectors that satisfy the constraints

Almost Every Problem can be posed as an Optimization Problem

- Given a set $\mathcal{C} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{C} \subseteq \mathcal{E}$.
  Hint: First work out the problem in lower dimensions



x in C is a vector of size n
x = [x1,x2.....xn]



NEED A ROTATED+TRANSLATED VERSION

Constraint: x1^2/a1^2 + x2^2/a2^2
+ ....xn^2/an^2 <= 1

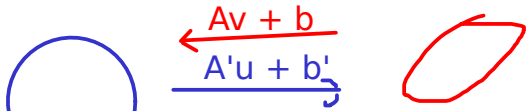
Almost Every Problem can be posed as an Optimization Problem

- Given a set $\mathcal{C} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{C} \subseteq \mathcal{E}$. *Hint: First work out the problem in lower dimensions*
- Sphere $\mathcal{S}_r \subseteq \Re^n$ centered at 0 is expressed as:

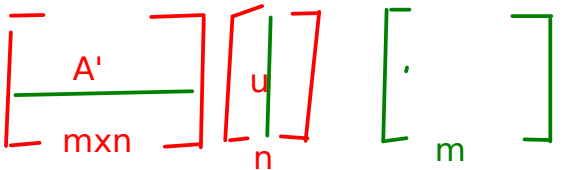
Sr = { ||x||_2 <= r}
2-norm is the square root of sum of squares of the individual components of x

Almost Every Problem can be posed as an Optimization Problem

- Given a set $\mathcal{C} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{C} \subseteq \mathcal{E}$. *Hint: First work out the problem in lower dimensions*
- Sphere $\mathcal{S}_r \subseteq \Re^n$ centered at 0 is expressed as: $\mathcal{S}_r = \{\mathbf{u} \in \Re^n | \|\mathbf{u}\|_2 \leq r\}$
- Ellipsoid $\mathcal{E} \subseteq \Re^n$ is expressed as:



Ellipsoid is a rotated, scaled and translated version of the sphere



Our basic ellipsoid had A' = diagonal

Almost Every Problem can be posed as an Optimization Problem

- Given a set $\mathcal{C} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{C} \subseteq \mathcal{E}$. *Hint: First work out the problem in lower dimensions*
- Sphere $\mathcal{S}_r \subseteq \Re^n$ centered at 0 is expressed as: $\mathcal{S}_r = \{\mathbf{u} \in \Re^n | \|\mathbf{u}\|_2 \leq r\}$
- Ellipsoid $\mathcal{E} \subseteq \Re^n$ is expressed as: $\mathcal{E} = \left\{\mathbf{v} \in \Re^n | A\mathbf{v} + \mathbf{b} \in \mathcal{S}_1\right\} = \left\{\mathbf{v} \in \Re^n | \|A\mathbf{v} + \mathbf{b}\|_2 \leq 1\right\}$. Here, $A \in \mathcal{S}^n_{++}$, that is, A is an $n \times n$ (strictly) positive definite matrix.
- The optimization problem will be:

1) A is an nxn matrix (Sphere and Ellipsoid are both in R^n)
2) This brings an additional constraint that A is symmetic, and it is positive definite

3) That is, A has positive eigen values..
4) The positive eigenvalues will correspond to scaling of the axis and corresponding eigenvectors to the new axes
5) The volume is proportional to the product of lengths of eigenvalues

Almost Every Problem can be posed as an Optimization Problem

- Given a set $\mathcal{C} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{C} \subseteq \mathcal{E}$. *Hint: First work out the problem in lower dimensions*
- Sphere $\mathcal{S}_r \subseteq \Re^n$ centered at 0 is expressed as: $\mathcal{S}_r = \{\mathbf{u} \in \Re^n | \|\mathbf{u}\|_2 \leq r\}$
- Ellipsoid $\mathcal{E} \subseteq \Re^n$ is expressed as: $\mathcal{E} = \left\{\mathbf{v} \in \Re^n | A\mathbf{v} + \mathbf{b} \in \mathcal{S}_1\right\} = \left\{\mathbf{v} \in \Re^n | \|A\mathbf{v} + \mathbf{b}\|_2 \leq 1\right\}$. Here, $A \in \mathcal{S}^n_{++}$, that is, A is an $n \times n$ (strictly) positive definite matrix.
- The optimization problem will be:

$$\underset{[a_{11},a_{12}..,a_{nn},b_1,..b_n]}{\text{minimize}} \quad det(A^{-1})$$

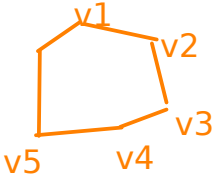
subject to $\quad \mathbf{v}^T A\mathbf{v} > 0, \ \forall \mathbf{v} \neq 0$ — A is positive definite

$\|A\mathbf{v} + \mathbf{b}\|_2 \leq 1, \ \forall \mathbf{v} \in \mathcal{C}$ — C is contained in the Ellipsoid

Can forall v be changed to checking for a finite number of boundary points?

Almost Every Problem can be posed as an Optimization Problem (contd.)

- Given a polygon $\mathcal{P} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{P} \subseteq \mathcal{E}$.
- Let $\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_p$ be the corners of the polygon $\mathcal{P}$
- The optimization problem will be:



$$\begin{array}{ll} \underset{[a_{11},a_{12}..,a_{nn},b_1,..b_n]}{\text{minimize}} & det(A^{-1}) \\ \text{subject to} & -\mathbf{v}^T A\mathbf{v} > 0, \ \forall \mathbf{v} \neq 0 \\ & \|A\mathbf{v}_i + \mathbf{b}\|_2 \leq 1, \ i \in \{1..p\} \end{array}$$

Given that the specified set S is indeed a polygon, is this problem with a simplified set of constraints equivalent to the original problem?
YES

Almost Every Problem can be posed as an Optimization Problem (contd.)

- Given a polygon $\mathcal{P} \subseteq \Re^n$ find the ellipsoid $\mathcal{E} \subseteq \Re^n$ that is of smallest volume such that $\mathcal{P} \subseteq \mathcal{E}$.
- Let $\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_p$ be the corners of the polygon $\mathcal{P}$
- The optimization problem will be:

$$\begin{aligned} \underset{[a_{11},a_{12}..,a_{nn},b_1,..b_n]}{\text{minimize}} \quad & det(A^{-1}) \\ \text{subject to} \quad & -\mathbf{v}^T A\mathbf{v} > 0, \ \forall \mathbf{v} \neq 0 \\ & \|A\mathbf{v}_i + \mathbf{b}\|_2 \leq 1, \ i \in \{1..p\} \end{aligned}$$

- How would you pose an optimization problem to find the ellipsoid $\mathcal{E}'$ of largest volume that fits inside $\mathcal{C}$?

So Again: Mathematical optimization

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \ i = 1, \ldots, m. \end{aligned}$$

$x = (x_1, ..., x_n)$: optimization variables
$f_i : \Re^n \to \Re, \ i = 1, ..., m$: constraint functions
optimal solution x^* has smallest value of f_0 among all vectors that satisfy the constraints

Examples

portfolio optimization

- variables: amounts invested in different assets
- constraints: budget, max./min. investment per asset, minimum return
- objective: overall risk or return variance

Examples

device sizing in electronic circuits

- variables: device widths and lengths
- constraints: manufacturing limits, timing requirements, maximum area
- objective: power consumption

Examples

Data fitting - Machine learning

- variables: model parameters
- constraints: prior information, parameter limits
- objective: measure of misfit or prediction error

An Example in Classification

Motivation

- Human beings better in categorical judgements than absolute scoring

Motivation

- Human beings better in categorical judgements than absolute scoring
- Classification vs. regression
  - MidSem: Good or Bad vs. Absolute score
  - Course register: Yes or No vs. how much(?)

Applications

- Spam Detection
- Digit Recognition
- Medical diagnosis
- Bio-diversity classification
- Buying or selling products

Problem in Perspective

- Given data points $x_i, i = 1, 2, \ldots, m$
- Possible class choices: $c_1, c_2, \ldots, c_k$
- Wish to generate a mapping/classifier

$$f : x \to \{c_1, c_2, \ldots, c_k\}$$

Problem in Perspective

- Given data points $x_i, i = 1, 2, \ldots, m$
- Possible class choices: $c_1, c_2, \ldots, c_k$
- Wish to generate a mapping/classifier

$$f : x \rightarrow \{c_1, c_2, \ldots, c_k\}$$

- To get class labels $y_1, y_2, \ldots, y_m$

Problem in Perspective

- In general, series of mappings

$$x \xrightarrow{f(\cdot)} y$$

Problem in Perspective

- In general, series of mappings

$$x \xrightarrow{f(\cdot)} y \xrightarrow{g(\cdot)} z \xrightarrow{h(\cdot)} \{c_1, c_2, \ldots, c_k\}$$

eg: neural networks

Problem in Perspective

- In general, series of mappings

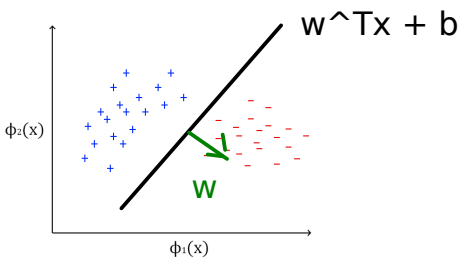
$$x \xrightarrow{f(\cdot)} y \xrightarrow{g(\cdot)} z \xrightarrow{h(\cdot)} \{c_1, c_2, \ldots, c_k\}$$

where y, z are in some latent space.

Binary Classification using Perceptron

Perceptron Classifier

- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$

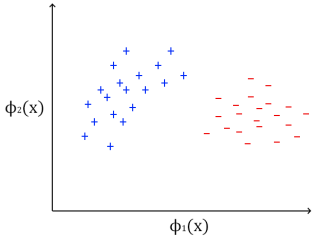


- Objective: Learn a linear classifier
- With linearly separability

Extent of misclassification of a point is = -y(w^Tx + b)

Perceptron Classifier

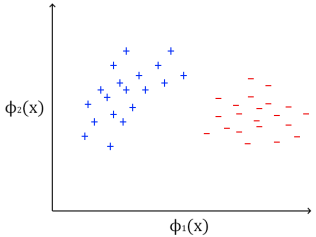
- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$



- Objective: Learn a linear classifier
- With linearly separability, any finite time learning algorithm?

Perceptron Classifier

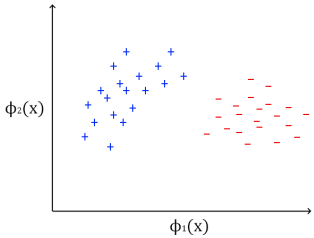
- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$



- Objective: Learn a linear classifier
- With linearly separability, any finite time learning algorithm?

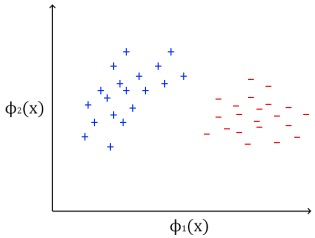
Perceptron Classifier

- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Desirable: Any new input pattern similar to a seen pattern is **classified** correctly

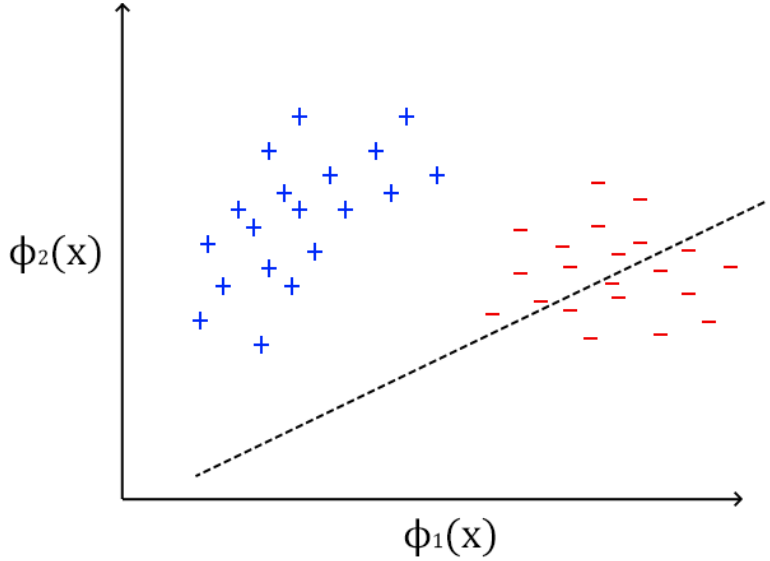


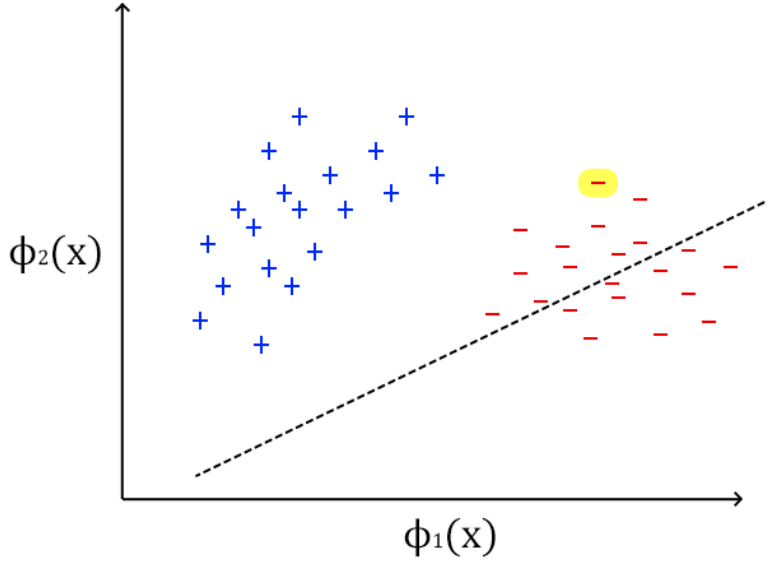
Perceptron Classifier

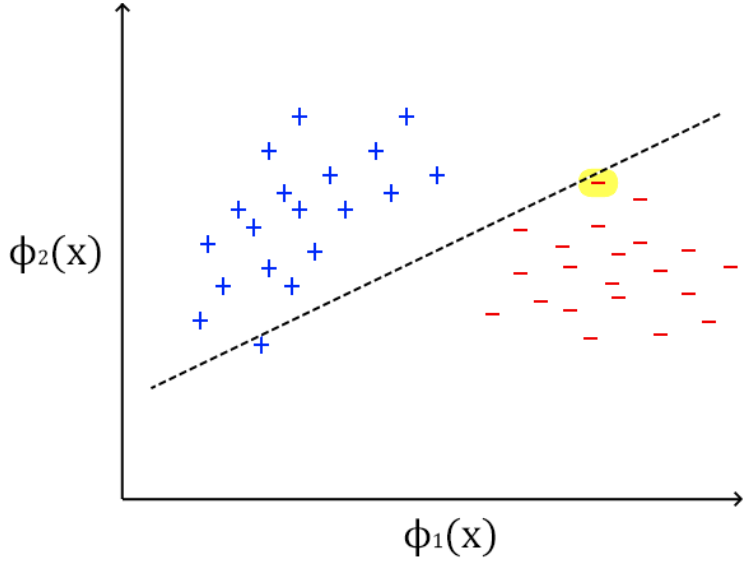
- Consider a binary classification problem: $f(\mathbf{x}) \in \{-1, +1\}$
- Desirable: Any new input pattern similar to a seen pattern is **classified** correctly

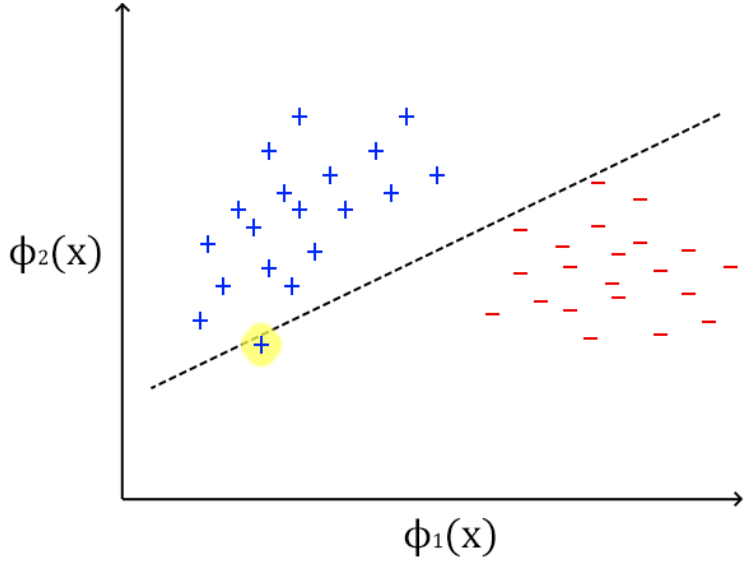


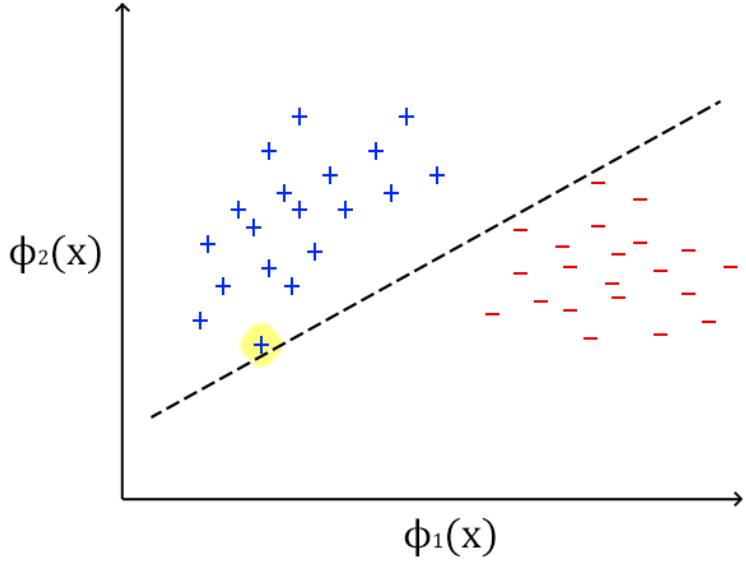
Linear Classification?
$\mathbf{w}^\top \phi(\mathbf{x}) + b \geq 0$ *for +ve points ($y = +1$)*
$\mathbf{w}^\top \phi(\mathbf{x}) + b < 0$ *for -ve points (y= -1)*
$\mathbf{w}, \phi \in \mathbb{R}^m$


$\phi_2(x)$
$\phi_1(x)$


$\phi_2(x)$
$\phi_1(x)$


$\phi_2(x)$
$\phi_1(x)$


$\phi_2(x)$
$\phi_1(x)$


$\phi_2(x)$
$\phi_1(x)$

Perceptron Update Rule: Error Perspective

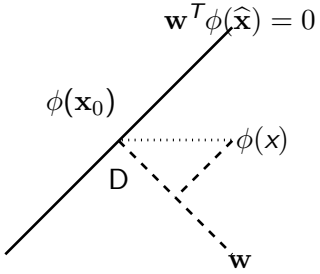
- Explicitly account for signed distance of (misclassified) points from the hyperplane $\mathbf{w}^T\phi(\widehat{\mathbf{x}}) = 0$.
  Consider point $\mathbf{x}_0$ with $\mathbf{w}^T(\phi(\mathbf{x}_0)) = 0$
  Signed distance from hyperplane: $\mathbf{w}^T(\phi(\mathbf{x}) - \phi(\mathbf{x}_0)) = \mathbf{w}^T\phi(\mathbf{x})$

Perceptron Update Rule: Error Perspective

- Explicitly account for signed distance of (misclassified) points from the hyperplane $\mathbf{w}^T\phi(\widehat{\mathbf{x}}) = 0$.
  Consider point $\mathbf{x}_0$ with $\mathbf{w}^T(\phi(\mathbf{x}_0)) = 0$
  Signed distance from hyperplane: $\mathbf{w}^T(\phi(\mathbf{x}) - \phi(\mathbf{x}_0)) = \mathbf{w}^T\phi(\mathbf{x})$

Perceptron Update Rule: Error Perspective

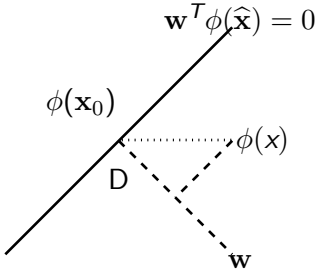
- Unsigned distance from hyperplane: $y\mathbf{w}^T(\phi(\mathbf{x}))$



Negative of the unsigned distance is the error

Perceptron Update Rule: Error Perspective

- Unsigned distance from hyperplane: $y\mathbf{w}^T(\phi(\mathbf{x}))$



- If $\mathbf{x}$ is misclassified, the misclassification cost for $\mathbf{x}$ is $-y\mathbf{w}^T(\phi(\mathbf{x}))$

Perceptron Update Rule: Error Minimization

- Perceptron update tries to minimize the error function
  E = negative of sum of unsigned distances over misclassified examples = sum of misclassification costs

$$E = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\mathbf{w}^T\phi(\mathbf{x})$$

Perceptron Update Rule: Error Minimization

- Perceptron update tries to minimize the error function
  E = negative of sum of unsigned distances over misclassified examples = sum of misclassification costs

$$E = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\mathbf{w}^T\phi(\mathbf{x})$$

  where $\mathcal{M} \subseteq \mathcal{D}$ is the set of misclassified examples.

Machine Learning as Optimization

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\text{argmin}} \; \mathcal{L}(\mathbf{w}) + \Omega(\|\mathbf{w}\|_2) \tag{1}$$

- **0-1 Loss:**

$$\mathcal{L}(\mathbf{w}) = \sum_{(\mathbf{x},y)} \delta\left(y \neq \mathbf{w}^T \phi(\mathbf{x})\right) \tag{2}$$

Minimizing the 0-1 Loss is NP-hard. We therefore look for surrogates.

- **Perceptron:** A Non-convex Surrogate

$$\mathcal{L}(\mathbf{w}) = -\sum_{(\mathbf{x},y)\in\mathcal{M}} y\mathbf{w}^T \phi(\mathbf{x}) \tag{3}$$

where $\mathcal{M} \subseteq \mathcal{D}$ is the set of misclassified examples.

Convex Surrogates for 0-1 Loss in Classification

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \; \mathcal{L}(\mathbf{w}) + \Omega(\|\mathbf{w}\|_2) \tag{4}$$

- **Logistic Regression:**

$$\mathcal{L}(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1 + exp\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(i)}\right)\right)\right)\right)\right] \tag{5}$$

- **Sigmoidal Neural Net:**

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^L\left(\mathbf{x}^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right)\log\left(1 - \sigma_k^L\left(\mathbf{x}^{(i)}\right)\right)\right] \tag{6}$$

Convex Surrogates for 0-1 Loss in Classification

$$\widehat{\mathbf{w}}^* = \underset{\mathbf{w}}{\text{argmin}} \; \mathcal{L}(\mathbf{w}) + \Omega(\|\mathbf{w}\|_2) \tag{7}$$

- **Logistic Regression:**

$$\mathcal{L}(\mathbf{w}) = -\left[\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - \log\left(1 + exp\left(\mathbf{w}^T\phi\left(\mathbf{x}^{(i)}\right)\right)\right)\right)\right] \tag{8}$$

- **Sigmoidal Neural Net:**

$$\mathcal{L}(\mathbf{w}) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(\sigma_k^L\left(\mathbf{x}^{(i)}\right)\right) + \left(1 - y_k^{(i)}\right)\log\left(1 - \sigma_k^L\left(\mathbf{x}^{(i)}\right)\right)\right] \tag{9}$$

More Generally..

- x represents some action such as
  - portfolio decisions to be made
  - resources to be allocated
  - schedule to be created
  - vehicle/airline deflections
- Constraints impose conditions on outcome based on
  - performance requirements
  - manufacturing process
- Objective $f_0(x)$ might correspond to one of the following and should be desirably small
  - total cost
  - risk
  - negative profit

Solving optimization problems

General optimization problems

- very difficult to solve
- methods involve some compromise, e.g., very long computation time, or not always finding the solution

Exceptions: certain problem classes can be solved efficiently and reliably

- least-squares problems
- linear programming problems
- convex optimization problems

Least-squares

[OPTIONAL]

$$\underset{x}{\text{minimize}} \quad \|Ax - b\|_2^2$$

solving least-squares problems

- analytical solution: $x^* = (A^T A)^{-1} A^T b$
- reliable and efficient algorithms and software
- computation time proportional to $n^2 k$ ($A \in \mathbb{R}^{k \times n}$); less if structured
- a mature technology

using least-squares

- least-squares problems are easy to recognize
- a few standard techniques increase flexibility (e.g., including weights, adding regularization terms)

Linear programming [OPTIONAL]

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x \\ \text{subject to} & a_i^T x \geq b_i,\ i = 1, \ldots, m. \end{array}$$

solving linear programs

- no analytical formula for solution
- reliable and efficient algorithms and software
- computation time proportional to n^2m if $m \geq n$; less with structure
- a mature technology

using linear programs

- not as easy to recognize as least-squares problems
- a few standard tricks used to convert problems into linear programs (e.g., problems involving l_1- or l_∞-norms, piecewise-linear functions)

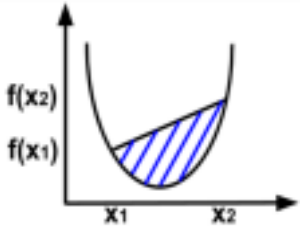
Convex optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f_0(x) \\ \text{subject to} & f_i(x) \leq b_i,\ i = 1, \ldots, m. \end{array}$$

- objective and constraint functions are convex:

$$f_i(\alpha x_1 + \beta x_2) \leq \alpha f_i(x_1) + \beta f_i(x_2)$$

if $\alpha + \beta = 1$, $\alpha \geq 0$, $\beta \geq 0$



- includes least-squares problems and linear programs as special cases

Convex optimization problem

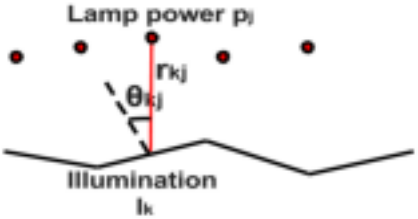
solving convex optimization problems

- no analytical solution
- reliable and efficient algorithms
- computation time (roughly) proportional to $\{n^3, n^2m, F\}$, where F is cost of evaluating f_i's and their first and second derivative
- almost a technology

using convex optimization

- often difficult to recognize
- many tricks for transforming problems into convex form
- surprisingly many problems can be solved via convex optimization

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]



intensity I_k at patch k depends linearly on lamp powers p_j:

$$I_k = \sum_{j=1}^{n} a_{kj} p_j,\ a_{kj} = {r_{kj}}^{-2} max\{cos\theta_{kj}, 0\}$$

problem: Provided the fixed locations(a_{kj}'s), achieve desired illumination I_{des} with bounded lamp powers

$$\begin{array}{ll} \underset{p_j}{\text{minimize}} & max_{k=1,..,n} \mid log(I_k) - log(I_{des}) \mid \\ \text{subject to} & 0 \leq p_j \leq p_{max},\ j = 1, \ldots, m. \end{array}$$

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]

How to solve? Some approximate(suboptimal) 'solutions':-

1. use uniform power: $p_j = p$, vary p
2. use least-squares:

$$\underset{p_j}{\text{minimize}} \quad \sum_{k=1}^{n} \|I_k - I_{des}\|_2^2$$

   round p_j if $p_j > p_{max}$ or $p_j < 0$
3. use weighted least-squares:

$$\underset{p_j}{\text{minimize}} \quad \sum_{k=1}^{n} \|I_k - I_{des}\|_2^2 + \sum_{j=1}^{m} w_j \|p_j - p_{max}/2\|_2^2$$

   iteratively adjust weights w_j until $0 \leq p_j \leq p_{max}$
4. use linear programming:

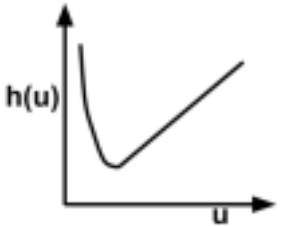
$$\begin{array}{ll} \text{minimize} & max_{k=1,..,n} \mid I_k - I_{des} \mid \\ \text{subject to} & 0 \leq p_j \leq p_{max},\ j = 1, \ldots, m. \end{array}$$

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]

- Use convex optimization: problem is equivalent to

$$\begin{array}{ll} \underset{p_j}{\text{minimize}} & f_0(p) = max_{k=1,..,n} h(I_k/I_{des}) \\ \text{subject to} & 0 \leq p_j \leq p_{max},\ j = 1, \ldots, m. \end{array}$$

with h(u) = max{u, 1/u}



- f_0 is convex because maximum of convex functions is convex
- **exact** solution obtained with effort $\approx$ modest factor $\times$ least-squares effort

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]

Additional constraints does adding 1 or 2 below complicate the problem?

1. no more than half of total power is in any 10 lamps.
2. no more than half of the lamps are on ($p_j > 0$).

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]

Additional constraints does adding 1 or 2 below complicate the problem?

1. no more than half of total power is in any 10 lamps.
2. no more than half of the lamps are on ($p_j > 0$).

- **answer:** with (1), still easy to solve; with (2), extremely difficult.

Example: m lamps illuminating n(small, flat) patches [OPTIONAL]

Additional constraints does adding 1 or 2 below complicate the problem?

1. no more than half of total power is in any 10 lamps.
2. no more than half of the lamps are on ($p_j > 0$).

- **answer:** with (1), still easy to solve; with (2), extremely difficult.
- **moral:** (untrained) intuition doesn't always work; without the proper background very easy problems can appear quite similar to very difficult problems.

Course goals and topics

Goals

- recognize/formulate problems (such as the illumination problem) as convex optimization problem
- develop code for problems of moderate size (1000 lamps, 5000 patches)
- characterize optimal solution (optimal power distribution), give limits of performance, etc

Topics

- Convex sets, (Univariate) Functions, Optimization problem
- Unconstrained Optimization: Analysis and Algorithms
- Constrained Optimization: Analysis and Algorithms
- Optimization Algorithms for Machine Learning
- Discrete Optimization and Convexity (Eg: Submodular Minimization)
- Other Examples and applications (MAP Inference on Graphical Models, Majorization-Minimization for Non-convex problems)

Nonlinear optimization

Traditional techniques for general nonconvex problems involve **Local optimization methods** (nonlinear programming)

- find a point that minimizes f_0 among feasible points near it
- fast, can handle large problems
- require initial guess
- provide no information about distance to (global) optimum

Global optimization methods

- find the (global) solution
- worst-case complexity grows exponentially with problem size

these algorithms are often based on solving convex subproblems

Grading and Audit

Grading

- Quizzes and Assignments: 15%
- Midsem: 25%
- Endsem: 45%
- Project: 15%

Audit requirement

- Quizzes and Assignments and Project

Brief history of convex optimization

theory (convex analysis): ca1900–1970

algorithms

- 1947: simplex algorithm for linear programming (Dantzig)
- 1960s: early interior-point methods (Fiacco & McCormick, Dikin, . . .)
- 1970s: ellipsoid method and other subgradient methods
- 1980s: polynomial-time interior-point methods for linear programming (Karmarkar 1984)
- late 1980s–now: polynomial-time interior-point methods for nonlinear convex optimization (Nesterov & Nemirovski 1994)

applications

- before 1990: mostly in operations research; few in engineering
- since 1990: many new applications in engineering (control, signal processing, communications, circuit design, . . .); new problem classes (semidefinite and second-order cone programming, robust optimization)