(Log) Barrier methods

Barrier Methods for Constrained Optimization

Consider a more general constrained optimization problem

$$\min_{\mathbf{x}\in\mathbf{R}^n} f(\mathbf{x})$$
$$\text{s.t.} g_i(\mathbf{x}) \leq 0 \; i = 1...m$$
$$\text{and } A\mathbf{x} = \mathbf{b}$$

Possibly reformulations of this problem include:

$$\min_x f(x) + \lambda B(x)$$

where B is a **barrier function** like

1. $B(x) = \frac{\rho}{2}\|A\mathbf{x} - \mathbf{b}\|^2$ (in Augmented Langragian - for a specific type of strong convexity wrt $\|.\|^2$))
2. $B(x) = \sum I_{g_i}(\mathbf{x})$ (Projected Gradient Descent: built on this & a linear approximation to $f(\mathbf{x})$)
3. $B(x) = \phi_{g_i}(\mathbf{x}) = -\frac{1}{t}\log\left(-g_i(\mathbf{x})\right)$
   - Here, $-\frac{1}{t}$ is used instead of λ. Lets discuss this in more details

Barrier Method: Example

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 primal) will be proved next.

Barrier Method and Linear Program

Recap:

Problem type	Objective Function	Constraints	$L^*(\lambda)$	Dual constraints	Strong duality
Linear Program	$\mathbf{c}^T\mathbf{x}$	$A\mathbf{x} \leq \mathbf{b}$	$-\mathbf{b}^T\lambda$	$A^T\lambda + \mathbf{c} = \mathbf{0}$	Feasible primal

What are necessary conditions at primal-dual optimality?

- ..
- ..

Log Barrier (Interior Point) Method

- The log barrier function is defined as

$$B(x) = \phi_{g_i}(\mathbf{x}) = -\frac{1}{t}\log\left(-g_i(\mathbf{x})\right)$$

- Approximates $\sum I_{g_i}(\mathbf{x})$ (better approximation as $t \to \infty$)
- $f(\mathbf{x}) + \sum_i \phi_{g_i}(\mathbf{x})$ is convex if f and g_i are convex
  Why? $\phi_{g_i}(\mathbf{x})$ is negative of monotonically increasing concave function (log) of a concave function $-g_i(\mathbf{x})$
- Let λ_i be lagrange multiplier associated with inequality constraint $g_i(\mathbf{x}) \leq 0$
- We've taken care of the inequality constraints, lets also consider an equality constraint $A\mathbf{x} = \mathbf{b}$ with corresponding langrage multipler (vector) ν

Log Barrier Method (contd.) (KKT based intepretation)

- Our objective becomes

$$\min_x f(x) + \sum_i \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$
$$\text{s.t. } Ax = b$$

- At different values of t, we get different $x^*(t)$
- Let $\lambda_i^*(t) =$
- First-order necessary conditions for optimality (and strong duality)[17] at $x^*(t), \lambda_i^*(t)$:
  1. ..
  2. ..
  3. ..
  4. ..
     - ..
-

[17] of original problem

- Our objective becomes

$$\min_x f(x) + \sum_i \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$
$$\text{s.t. } Ax = b$$

- At different values of t, we get different x^*
- Let $\lambda_i^*(t) = \frac{-1}{t\, g_i\left(x^*(t)\right)}$
- First-order necessary conditions for optimality (and strong duality)[18] at $x^*(t), \lambda_i^*(t)$:
  1. $g_i\left(x^*(t)\right) \leq 0$
  2. $Ax^*(t) = b$
  3. $\nabla f\left(x^*(t)\right) + \sum_{i=1}^m \lambda_i^*(t) \nabla g_i\left(x^*(t)\right) + \nu^*(t)^\top A = 0$
  4. $\lambda_i^*(t) \geq 0$
     - Since $g_i\left(x^*(t)\right) \leq 0$ and $t \geq 0$
- All above conditions hold at optimal solution $\mathbf{x}(t), \nu(t)$, of barrier problem $\Rightarrow$ $\left(\lambda_i^*(t), \nu^*(t)\right)$ are dual feasible. (onlt complementary slackness is violated)

[18] of original problem

Log Barrier Method & Duality Gap (KKT based intepretation)

- If necessary conditions are satisfied and **if f and g_i's are convex, and g_i's strictly feasible**, the conditions are also sufficient. Thus, $\left(x^*(t), \lambda_i^*(t), \nu^*(t)\right)$ form a critical point for the Lagrangian

$$L(\mathbf{x}, \lambda, \nu) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \nu^\top (A\mathbf{x} - \mathbf{b})$$

- Lagrange dual function

$$L^*(\lambda, \nu) = \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \nu)$$

$$L^*\left(\lambda^*(t), \nu^*(t)\right) = f\left(\mathbf{x}^*(t)\right) + \sum_{i=1}^{m} \lambda_i^*(t) g_i\left(\mathbf{x}^*(t)\right) + \nu^*(t)^\top \left(A\mathbf{x}^*(t) - \mathbf{b}\right)$$

$$= f(x^*(t)) - m/t$$

  - m/t ...... is the *duality gap* upperbound
  - As $t \to \infty$, duality gap $\to$. 0

Log Barrier Method & Duality Gap (KKT based intepretation)

- If necessary conditions are satisfied and **if f and g_i's are convex, and g_i's strictly feasible**, the conditions are also sufficient. Thus, $\left(x^*(t), \lambda_i^*(t), \nu^*(t)\right)$ form a critical point for the Lagrangian

$$L(\mathbf{x}, \lambda, \nu) = f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) + \nu^\top (A\mathbf{x} - \mathbf{b})$$

- Lagrange dual function

$$L^*(\lambda, \nu) = \min_{\mathbf{x}} L(\mathbf{x}, \lambda, \nu)$$

$$L^*\left(\lambda^*(t), \nu^*(t)\right) = f\left(\mathbf{x}^*(t)\right) + \sum_{i=1}^{m} \lambda_i^*(t) g_i\left(\mathbf{x}^*(t)\right) + \nu^*(t)^\top \left(A\mathbf{x}^*(t) - \mathbf{b}\right)$$

$$= f\left(x^*(t)\right) - \frac{m}{t}$$

  - $\frac{m}{t}$ here is called the *duality gap*
  - As $t \to \infty$, duality gap $\to 0$, but computing optimal solution $\mathbf{x}(t)$ to barrier problem will be that harder

Log Barrier Method & Duality Gap (KKT based intepretation)

- At optimality, primal optimal = dual optimal
  i.e. $p^* = d^*$
- From weak duality,

$$f\left(\mathbf{x}^*(t)\right) - \frac{m}{t} \leq p^*$$

$$\implies f\left(\mathbf{x}^*(t)\right) - p^* \leq \frac{m}{t}$$

  - The duality gap is always $\leq \frac{m}{t}$
  - The more we increase t, the smaller will be the duality gap

Log Barrier method: Start with small t (conservative about feasibility set
Iteratively solve the barrier formulation (start with solution to prev iterat
increase value of t

The Log Barrier Method

Also known as sequential unconstrained minimization technique (SUMT) & barrier method & path-following method

1. Start with $t = t^{(0)}$, $\mu > 1$, and consider ϵ tolerance
2. **Repeat**

   INNER ITERATION: (solved using Dual Ascent or Augment Lagrangian)

   Newton algo especially good for this

   1. **Solve**

      $$\mathbf{x}^*(t) = \underset{x}{\text{argmin}}\ f(\mathbf{x}) + \sum_{i=1}^{m} \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

      $$\text{s.t. } Ax = b$$

      for solving for x*(t), initialize using x*(t-1)

   2. If $\frac{m}{t} < \epsilon$, **Quit**

      else, **set** $t = \mu t$

Scale up the value of t multiplicatively in every outer iteration

The Log Barrier Method

Also known as sequential unconstrained minimization technique (SUMT) & barrier method & path-following method

1. Start with $t = t^{(0)}$, $\mu > 1$, and consider ϵ tolerance
2. **Repeat**
   1. **Solve**

$$\mathbf{x}^*(t) = \underset{x}{\text{argmin}}\, f(\mathbf{x}) + \sum_{i=1}^{m} \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

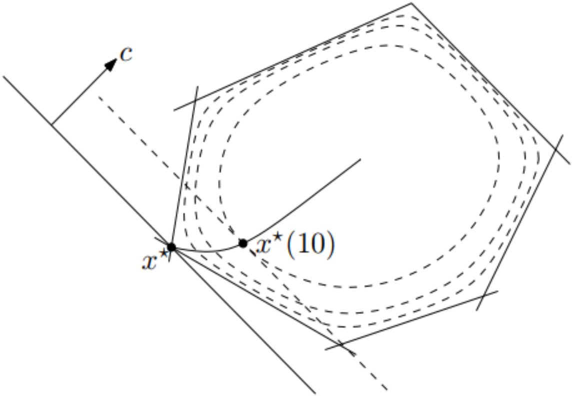
$$\text{s.t. } Ax = b$$

   2. If $\frac{m}{t} < \epsilon$, **Quit**
      else, **set** $t = \mu t$

Note: Computing $\mathbf{x}^*(t)$ exactly is not necessary since the central path has no significance other than that it leads to a solution of the original problem for $t \to \infty$;
Also small $\mu \Rightarrow$ faster inner iterations. Large $\mu \Rightarrow$ faster outer iterations.

Since x*(t-1) will not be far from x*(t)

Upper bound on duality gap will shrink quickly



Central path for an LP with n = 2 and m = 6. The dashed curves show three contour lines of the logarithmic barrier function φ. The central path converges to the optimal point x* as t → ∞. Also shown is the point on the central path with t = 10.
[Figure source: Boyd & Vandenberghe]

- In the process, we can also obtain $\lambda^*(t)$ and $\nu^*(t)$
- **Convergence of outer iterations:**

  We get ϵ accuracy after $\left(\frac{\log\left(\frac{m}{\epsilon t^{(0)}}\right)}{log(\mu)}\right)$ updates of t

Log Barrier Method & Strictly Feasible Starting Point

- The inner optimization in the iterative algorithm using a barrier method,

$$\mathbf{x}^*(t) = \operatorname*{argmin}_x f(x) + \sum_i \left(-\frac{1}{t}\right) \log\left(-g_i(x)\right)$$

$$\text{s.t. } A\mathbf{x} = b$$

  can be solved using (sub)gradient descent starting from older value of x from previous iteration

- We must start with a strictly feasible $\mathbf{x}$, otherwise $-\log\left(-g_i(\mathbf{x})\right) \to \infty$

How to find a strictly feasible $\mathbf{x}^{(0)}$?

How to find a strictly feasible $\mathbf{x}^{(0)}$?

- *Basic Phase I method*

$$\mathbf{x}^{(0)} = \operatorname*{argmin}_{\mathbf{x}} \Gamma$$

$$\text{s.t. } g_i(\mathbf{x}) \leq \Gamma$$

- We solve this using the barrier method, and thus will also need a strictly feasible starting $\hat{\mathbf{x}}^{(0)}$
- Here,

$$\Gamma = \max_{i=1\ldots m} g_i(\hat{\mathbf{x}}^{(0)}) + \delta$$

  where, $\delta > 0$
  - *i.e.* Γ is slightly larger than the largest $g_i(\hat{\mathbf{x}}^{(0)})$

- On solving this optimization for finding $\mathbf{x}^{(0)}$,
  - If $\Gamma^* < 0$, $\mathbf{x}^{(0)}$ is strictly feasible
  - If $\Gamma^* = 0$, $\mathbf{x}^{(0)}$ is feasible (but not strictly)
  - If $\Gamma^* > 0$, $\mathbf{x}^{(0)}$ is not feasible
- A slightly 'richer' problem can consider different Γ_i for each g_i, to improve numerical precision

$$\boxed{\begin{aligned} \mathbf{x}^{(0)} &= \underset{\mathbf{x}}{\text{argmin}}\, \Gamma_i \\ \text{s.t. } g_i(\mathbf{x}) &\leq \Gamma_i \end{aligned}}$$

min over i

Choice of a good $\hat{\mathbf{x}}^{(0)}$ or $\mathbf{x}^{(0)}$ depends on the nature/class of the problem, use domain knowledge to decide it

Log Barrier Method & Strictly Feasible Starting Point

- We need not obtain $\mathbf{x}^*(t)$ exactly from each outer iteration
- If not solving for $\mathbf{x}^*(t)$ exactly, we will get ϵ accuracy after *more than* $\left(\frac{\log\left(\frac{m}{\epsilon t^{(0)}}\right)}{\log(\mu)}\right)$ updates of t
  - However, solving the inner iteration exactly may take too much time
  - Fewer inner loop iterations correspond to more outer loop iterations

TRADEOFFS

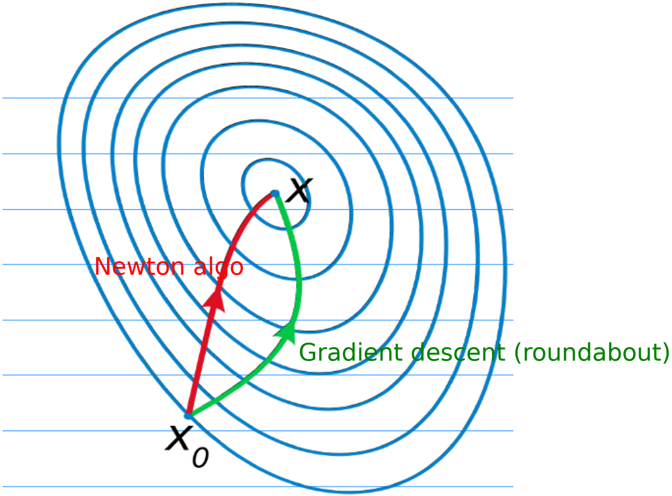
Log Barrier Method & Strictly Feasible Starting Point

- We need not obtain $\mathbf{x}^*(t)$ exactly from each outer iteration
- If not solving for $\mathbf{x}^*(t)$ exactly, we will get ϵ accuracy after *more than* $\left(\frac{\log\left(\frac{m}{\epsilon t^{(0)}}\right)}{\log(\mu)}\right)$ updates of t
  - However, solving the inner iteration exactly may take too much time
  - Fewer inner loop iterations correspond to more outer loop iterations
- Second order descent algorithms (such as Newton Descent) found effective in such settings for following reasons:

Log Barrier Method & Strictly Feasible Starting Point

- We need not obtain $\mathbf{x}^*(t)$ exactly from each outer iteration
- If not solving for $\mathbf{x}^*(t)$ exactly, we will get ϵ accuracy after *more than* $\left(\frac{\log\left(\frac{m}{\epsilon t^{(0)}}\right)}{\log(\mu)} \right)$ updates of t
  - However, solving the inner iteration exactly may take too much time
  - Fewer inner loop iterations correspond to more outer loop iterations
- Second order descent algorithms (such as Newton Descent) found effective in such settings for following reasons:
  Recall: Curvature naturally characterized by the Hessian
  - Accounts for curvature of the function; useful to converge to $\mathbf{x}(\mu t)$ quickly from $\mathbf{x}(t)$.
  - Quadratic convergence when close to $\mathbf{x}^*(t)$ Proved in Boyd
  - Less (or no) dependence on step size t^k Accouting for curvature reduces sensitivity to step size

Second Order Descent and Approximations
Sections 4.5.2 - 4.5.6 of BasicsOfConvexOptimization.pdf


x
Newton algo
Gradient descent (roundabout)
x
0

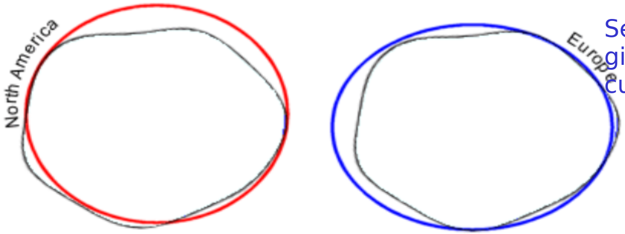
Newton's Algorithm as a Steepest Descent Method

- This choice of $\Delta \mathbf{x}^{k+1}$ corresponds to the direction of steepest descent under the matrix norm[19] induced by the Hessian $\nabla^2 f(\mathbf{x}^k)$:
  $\Delta \mathbf{x}^{(k)} = \text{argmin } \left\{ \nabla^T f(\mathbf{x}^{(k)})\mathbf{v} \mid ||\mathbf{v}||_{\nabla^2 f(\mathbf{x}^k)} = 1 \right\}.$
- Equivalently, based on approximating a function around the current iterate $\mathbf{x}^{(k)}$ using a second degree Taylor expansion.

$$Q(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)})$$

- Convex $f \Rightarrow$ Hessian is positive semi-definite
  Q(x) will ALSO be convex

[19] $\left(\mathbf{v}^T \nabla^2 f(\mathbf{x}^k)\mathbf{v}\right)^{\frac{1}{2}}$

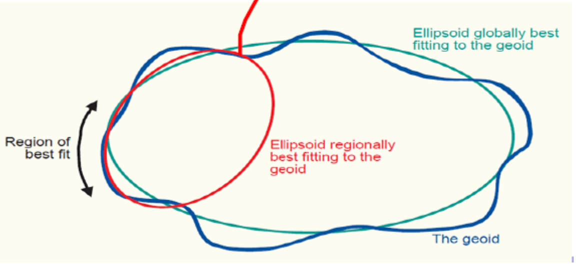


The red ellipsoid fits the geoid well in North America.

The blue ellipsoid fits the geoid well in Europe.

Second order information through hessian gives an ellipsoid which is aligned with the curvature in that specific region (Europe)

Local quadratic approximation to surface



Newton's Algorithm as a Steepest Descent Method

- This choice of $\Delta \mathbf{x}^{k+1}$ corresponds to the direction of steepest descent under the matrix norm[19] induced by the Hessian $\nabla^2 f(\mathbf{x}^k)$:
  $\Delta \mathbf{x}^{(k)} = \text{argmin} \left\{ \nabla^T f(\mathbf{x}^{(k)}) \mathbf{v} \mid ||\mathbf{v}||_{\nabla^2 f(\mathbf{x}^k)} = 1 \right\}$.
- Equivalently, based on approximating a function around the current iterate $\mathbf{x}^{(k)}$ using a second degree Taylor expansion.

$$Q(\mathbf{x}) \approx \widetilde{f}(\mathbf{x}) = f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)})$$

- Convex $f \Rightarrow$ convex quadratic approximation. Newton's method is based on solving the approximation exactly
- Setting gradient of quadratic approximation (with respect to $\mathbf{x}$) to $\mathbf{0}$ gives

$$\nabla^T f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)})(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = 0$$

Assuming $\nabla^2 f(\mathbf{x}^k)$ is invertible, next iterate is $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)})$

[19] $\left(\mathbf{v}^T \nabla^2 f(\mathbf{x}^k) \mathbf{v}\right)^{\frac{1}{2}}$

Newton's Algorithm as a Steepest Descent Method

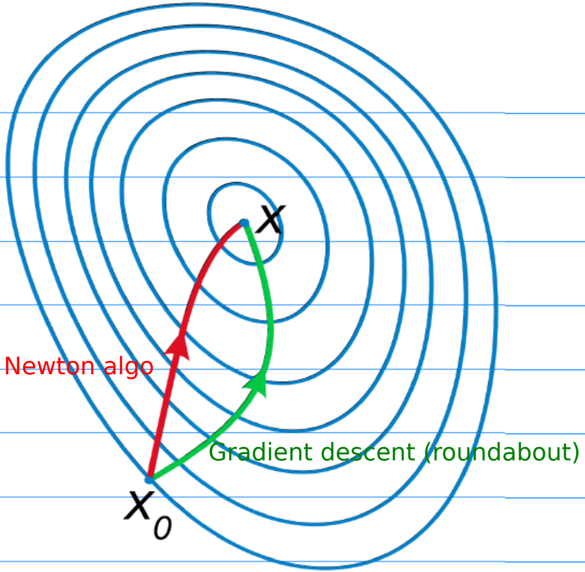
Find a starting point $\mathbf{x}^{(0)} \in \mathcal{D}$.
Select an appropriate tolerance $\epsilon > 0$.
repeat

1. Set $\Delta\mathbf{x}^{(k)} = -\left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x})$.
2. Let $\lambda^2 = \nabla^T f(\mathbf{x}^{(k)}) \left(\nabla^2 f(\mathbf{x}^{(k)})\right)^{-1} \nabla f(\mathbf{x}^{(k)}) \Leftrightarrow$ Directional derivative in the Newton Direction
3. If $\frac{\lambda^2}{2} \leq \epsilon$, **quit**.
4. Set step size $t^{(k)} = 1$. Obtain $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + t^{(k)}\Delta\mathbf{x}^{(k)}$.
5. Set $k = k + 1$.

until

Figure 32: The Newton's method which typically uses a step size of 1. $\Delta\mathbf{x}^{(k)}$ can be shown to be always a Descent Direction (Theorem 83 of notes). For $\mathbf{x} \in \Re^n$, each Newton's step takes $O(n^3)$ time (without using any fast matrix multiplication methods).

Most expensive step: Computing Hessian inverse


x
Newton algo
Gradient descent (roundabout)
x_0

Variants of Newtons's Method

- **Special Cases:** When Objective function is a composition of two functions (such as Loss l over some Prediction function m): Gauss Newton Approximation (Section 4.5.4 of BasicsOfConvexOptimization.pdf) and Levenberg-Marquardt (Section 4.5.5)
- **Quasi-Newton Algorithms:** When Hessian inverse $\left(\nabla^2 f(\mathbf{x}^{k+1})\right)^{-1}$ is approximated by a matrix B^{k+1} such that
  - gradient of quadratic approximation $Q(\mathbf{x}^k)$ agrees at $\mathbf{x}^k$ and $\mathbf{x}^{k+1}$
  - B^{k+1} is as close as possible to B^k in some norm (such as the Frobenius norm)

  See BFGS (Section 4.5.6), LBFGS *etc.*

Cutting Plane Algorithm
(Invoking Linear Programs for Non-linear constraints)

Cutting Plane Algorithm

Consider amother general formulation of convex optimization problems[20]:

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & g_j(\mathbf{x}) \leq 0 \quad \text{for } j = 1, 2, \ldots, m \end{array} \tag{92}$$

where $g_j(\mathbf{x})$ are convex functions.

- How can every convex optimization problem be presented in this form?

[20] All convex optimization problems of the form discussed so far can be cast in this form.

Cutting Plane Algorithm

Consider amother general formulation of convex optimization problems[20]:

$$\begin{aligned} &\text{minimize} && \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && g_j(\mathbf{x}) \leq 0 \quad \text{for } j = 1, 2, \ldots, m \end{aligned} \tag{92}$$

where $g_j(\mathbf{x})$ are convex functions.

- How can every convex optimization problem be presented in this form? For objective function $f(\mathbf{x})$, translate it into a constraint $f(\mathbf{x}) - c \leq 0$ and minimize c
- Let $\mathbf{s}_j(\mathbf{x}^i)$ be a subgradient for g_j at $\mathbf{x}^i$. By definition of subgradient

[20] All convex optimization problems of the form discussed so far can be cast in this form.

Cutting Plane Algorithm

Consider amother general formulation of convex optimization problems[20]:

$$\begin{aligned} &\text{minimize} && \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && g_j(\mathbf{x}) \le 0 \quad \text{for } j = 1, 2, \ldots, m \end{aligned} \tag{92}$$

where $g_j(\mathbf{x})$ are convex functions.

- How can every convex optimization problem be presented in this form? For objective function $f(\mathbf{x})$, translate it into a constraint $f(\mathbf{x}) - c \le 0$ and minimize c
- Let $\mathbf{s}_j(\mathbf{x}^i)$ be a subgradient for g_j at $\mathbf{x}^i$. By definition of subgradient $g_j(\mathbf{x}) \ge g_j(\mathbf{x}^i) + \mathbf{s}_j^T(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i)$ for all $\mathbf{x} \in dom(g_j)$. [Eg: $\mathbf{s}_j(\mathbf{x}^i)$ could be $\nabla g_j(\mathbf{x}^i)$]

[20] All convex optimization problems of the form discussed so far can be cast in this form.

Cutting Plane Algorithm (contd.)

- Since we are restricting the search to $\mathbf{x}$ such that $g_j(\mathbf{x}) \leq 0$,

The tangent hyperplane lower bound to g_j shopuld necessarily be also <= 0

Cutting Plane Algorithm (contd.)

- Since we are restricting the search to $\mathbf{x}$ such that $g_j(\mathbf{x}) \leq 0$, $0 \geq g_j(\mathbf{x}^i) + \mathbf{s}_j^T(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i)$ for all $\mathbf{x} \in dom(g_j)$
- When the last inequality is enumerated for all values of i and j, we get several linear constraints:
  $\mathbf{s}_j^T(\mathbf{x}^i)\mathbf{x} \leq \mathbf{s}_j^T(\mathbf{x}^i)\mathbf{x}^i - g_j(\mathbf{x}^i)$ for fixed i and all j and $\mathbf{x} \in dom(g_j) \equiv A_i\mathbf{x} \leq A_i\mathbf{x}^i - \mathbf{g}_i$

$$A_i = \begin{bmatrix} \mathbf{s}_1(\mathbf{x}^i) \\ \mathbf{s}_2(\mathbf{x}^i) \\ . \\ . \\ \mathbf{s}_m(\mathbf{x}^i) \end{bmatrix} \quad \mathbf{g}_i = \begin{bmatrix} g_1(\mathbf{x}^i) \\ g_2(\mathbf{x}^i) \\ . \\ . \\ g_m(\mathbf{x}^i) \end{bmatrix} \tag{93}$$

Cutting Plane Algorithm (contd.)

x^0 --> x^1 --> x^2... --> x^k

- Stacking all the A_i's and $\mathbf{g}_i$'s together

$$A^k = \begin{bmatrix} A_0 \\ A_1 \\ . \\ . \\ A_k \end{bmatrix} \quad \mathbf{b}^k = \begin{bmatrix} A_0\mathbf{x}^0 - \mathbf{g}_0 \\ A_1\mathbf{x}^1 - \mathbf{g}_1 \\ . \\ . \\ A_k\mathbf{x}^k - \mathbf{g}_k \end{bmatrix} \tag{94}$$

- With this, the necessary feasible conditions are: $A^k\mathbf{x} \leq \mathbf{b}^k$.
- Idea: Solve the following LP iteratively, until all original constraints are respected:

As k increases, number of constraints increases making it more and more likely that the original constraints are satisified

$$\begin{aligned} \mathbf{x}_*^k = \quad & \underset{\mathbf{x}}{\text{argmin}} \quad & \mathbf{c}^T\mathbf{x} \\ & \text{subject to} \quad & A^k\mathbf{x} \leq \mathbf{b}^k \end{aligned}$$

Kelly's Cutting Plane Algorithm (contd.)

Step 1
Input an initial feasible point, $\mathbf{x}^0$ and set $k = 0$.
Step 2: Evaluate A^k and $\mathbf{b}^k$
Step 3
Solve the LP problem

$$\mathbf{x}_*^k = \begin{array}{ll} \underset{\mathbf{x}}{\text{argmin}} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & A^k\mathbf{x} \leq \mathbf{b}^k \end{array}$$

Step 4
If $\underline{\max\{g_j(\mathbf{x}_*^k),\ 1 \leq j \leq m\} < \epsilon}$ output $\mathbf{x}_* = \mathbf{x}_*^k$ as the point of optimality and stop. Otherwise, set $k = k+1$, $\mathbf{x}^{k+1} = \mathbf{x}_*^k$, update A^k and $\mathbf{b}^k$ from (94) using (93) and repeat from **Step 3.**

Figure 33: Optimization for the convex problem in (92) using Kelly's cutting plane algorithm.

Primal Active-Set Algorithm (Lazy Projection Methods)

Interior point algo forced feasibility at every step
Projection methods force projection at every step
Active set==> Keep track of set of active and inactive constraints and be lazy in projection

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{95}$$

where $Q \succ 0$. The KKT conditions are:

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{95}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$...

Quadratic Optimization: Primal Active-Set Algorithm

$$
\begin{array}{ll}
\text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\
\text{subject to} & A\mathbf{x} \geq \mathbf{b}
\end{array} \tag{95}
$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If $\widehat{\mathbf{x}}$ lies in interior of feasible region then corresponding lambdas should be 0

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{95}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \cancel{\sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i} = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If $\widehat{\mathbf{x}}$ lies in interior of feasible region then
  1. $\widehat{\lambda} = 0$
  2. $\widehat{\mathbf{x}} = -Q^{-1}\mathbf{c}$

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{96}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\mathbf{a}_i^T\mathbf{x}^* = b_i$ for some $i \in I^*$ (index set of active constraints) then

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{96}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\mathbf{a}_i^T\mathbf{x}^* = b_i$ for some $i \in I^*$ (index set of active constraints) then, one needs to iteratively solve $\mathbf{x}^k$ and I_k

Basic idea: Assume that the only tests one needs to prepare for are the tests happening tomorrow (that is, the index set I_k) and that tests thereafter I_k complement) will be dealt with when one gets to them!!

Quadratic Optimization: Primal Active-Set Algorithm

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} + \beta \\ \text{subject to} & A\mathbf{x} \geq \mathbf{b} \end{array} \tag{96}$$

where $Q \succ 0$. The KKT conditions are:

- $Q\widehat{\mathbf{x}} + c - \sum_{i=1}^{m} \widehat{\lambda}_i \mathbf{a}_i = 0$
- $\widehat{\lambda}_i(\mathbf{a}_i^T\widehat{\mathbf{x}} - b_i) = 0$ for $i = 1..m$
- $\widehat{\lambda}_i \geq 0$ for $i = 1..m$
- $A\widehat{\mathbf{x}} \geq \mathbf{b}$... If some $\mathbf{a}_i^T\mathbf{x}^* = b_i$ for some $i \in I^*$ (index set of active constraints) then, one needs to iteratively solve $\mathbf{x}^k$ and I_k
  3. $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$
  4. Simplified objective: Find $\mathbf{d}^k = \underset{\mathbf{d}}{\text{argmin}}\ f_k(\mathbf{d})$

How much are we allowed to move so that the active constraints are not violated!

Quadratic Optimization: Primal Active-Set Algorithm

$$\mathbf{d}^k = \begin{array}{ll} \text{argmin} & f_k(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T Q\mathbf{d} + \mathbf{g}_k^T\mathbf{d} + c_k \\ \text{subject to} & \underline{\mathbf{a}_i\mathbf{d} = 0 \text{ for all } i \in I_k} \end{array} \tag{97}$$

where $\mathbf{g}_k = Q\mathbf{x}^k + \mathbf{c}$ and $c_k = (\mathbf{x}^k)^T Q\mathbf{x}^k + \mathbf{c}^T\mathbf{x}^k$. The idea behind the active set algo is:

1. $\mathbf{d}^k = 0 \Rightarrow \mathbf{x}^k$ satisfies first order necessary conditions:
   - $\mathbf{g}^k - \sum_{i \in I_k} \lambda_i \mathbf{a}_i = 0$ which is the same as $\underline{rank[A_{\mathcal{I}^k}^T \;\; \mathbf{g}^k] = rank[A_{\mathcal{I}^k}^T]}$

   We already know that $\mathbf{a}_i^T\mathbf{x}^k - b_i > 0\, \forall i \notin I_k$ and $\mathbf{a}_i^T\mathbf{x}^k - b_i = 0\, \forall i \in I_k$. Set $\lambda_i = 0\, \forall i \notin I_k$
   1. If $\lambda_i \geq 0\, \forall i \in I_k$, by KKT sufficient conditions, $\mathbf{x}^k$ will be point of global minimum.
   2. If $\lambda_i < 0$ for some $i \in I_k$, then it can be shown that if i is dropped from I_k, the active set and (97) is solved then $\mathbf{d}^k$ will be a descent direction $\nabla^T f(\mathbf{x}^k)\mathbf{d}^k < 0$ and reduce objective
2. $\mathbf{d}^k \neq 0 \;\Rightarrow$ we need to further determine α_k such that $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k\mathbf{d}^k$ remains

   feasible: $\alpha_k = \min\left\{1, \min_{\substack{j \notin \mathcal{I}^k \\ \mathbf{a}_j^T\mathbf{d}^k < 0}} \frac{\mathbf{a}_j^T\mathbf{x}^k - b_j}{-\mathbf{a}_j^T\mathbf{d}^k}\right\}$

Quadratic Optimization: Primal Active-Set Algorithm

Step 1

Input a feasible point, $\mathbf{x}^0$, identify the active set $\mathcal{I}^0$, form matrix $A_{\mathcal{I}^0}$, and set $k = 0$.

Step 2

Compute $\mathbf{g}^k = Q\mathbf{x}^k + \mathbf{c}$.

Check the rank condition $rank[A_{\mathcal{I}^k}^T \;\; \mathbf{g}^k] = rank[A_{\mathcal{I}^k}^T]$. If it does not hold, go to **Step 4**.

Step 3

Solve the system $A_{\mathcal{I}^k}^T \widehat{\lambda} = \mathbf{g}^k$. If $\widehat{\lambda} \geq \mathbf{0}$, output $\mathbf{x}^k$ as the solution and stop; otherwise, remove the index that is associated with the most negative Lagrange multiplier (some $\widehat{\lambda}_t$) from $\mathcal{I}^k$.

Step 4

Compute the value of $\mathbf{d}^k$:

$$\begin{aligned} \mathbf{d}^k = \quad & \underset{\mathbf{d}}{\text{argmin}} && \tfrac{1}{2}\mathbf{d}^T Q\mathbf{d} + (\mathbf{g}^k)^T\mathbf{d} \\ & \text{subject to} && \mathbf{a}_i^T\mathbf{d} = 0 \qquad \text{for } i \in \mathcal{I}^k \end{aligned} \tag{98}$$

Step 5

$$\alpha_k = \min \left\{ 1, \min_{\substack{j \notin \mathcal{I}^k \\ \mathbf{a}_j^T \mathbf{d}^k < 0}} \frac{\mathbf{a}_j^T \mathbf{x}^k - b_j}{-\mathbf{a}_j^T \mathbf{d}^k} \right\} \tag{99}$$

Set $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{d}^k$.

Step 6

If $\alpha_k < 1$, construct $\mathcal{I}^{k+1}$ by adding the index that yields the minimum value of α_k in (99). Otherwise, let $\mathcal{I}^{k+1} = \mathcal{I}^k$.

Step 7

Set $k = k + 1$ and repeat from **Step 2**.

Figure 34: Optimization for the quadratic problem in (96) using Primal Active-set Method.