Function Optimization

♣ Minimization or Maximization of Functions
♣ Steepest descent method
♣ conjugate gradient method
♣ Simulated annealing procedure
♣ Linear programming
◊ Other Topics with Applications
Minimization or Maximization of Functions

• Given a function $f$ defined on $\mathbb{R}^n$, find an $x \in \mathbb{R}^n$ to maximize or minimize $f(x)$

• A necessary (may not be sufficient) condition of $f$ achieving an optimal solution at $x_0$ is $\nabla f(x_0) = 0$

• Maximizing a function $f$ is equivalent to minimizing $-f$

Examples

[1] Find $x \in [-2, -2]$ to maximize or minimize $x^2 - 4x + 3$

[2] Find $(x, y)$ to minimize $f(x, y) = x^3 + y^3 - 2x^2 + 3y^2 - 8$
Elementary Gradient Methods

Goal: To minimize $f(x)$

Write $g(t) = f(x + tu)$, then $g'(0) = \nabla f(x)^t u$

General Gradient Minimization Algorithm

Given a smooth function $f(x)$ and an approximation $x^0$ to a (local) minimum $x^*$ of $f$.

Repeat

Define a search direction $u^{(i)}$

Find $t^*$ to minimize (or reduce) $g(t) = f(x^{(i)} + tu^{(i)})$

$x^{(i+1)} \leftarrow x^{(i)} + t^* u^{(i)}$

$i \leftarrow i + 1$

Until convergence criterion is satisfied.

/* The approximate minimum point is $x^{(i)}$ */
Steepest Descent Method

Goal: To minimize \( f(x) \)

\[
g(t) = f(x + tu) \Rightarrow g'(0) = \nabla f(x)^t u
\]

Steepest Descent Algorithm

Given a smooth function \( f(x) \) and an approximation \( x^{(0)} \) to a (local) minimum \( x^* \) of \( f \).

Repeat \( m = 0, 1, 2, \cdots \)

\[
u \equiv \nabla f(x^{(m)})
\]

if \( u = 0 \) return

else, determine the minimum \( t^* > 0 \) closest to 0 of the function

\[
g(t) = f(x^{(m)} - tu)
\]

\( x^{(m+1)} \leftarrow x^{(m)} - t^* u \)

Until convergence criterion is satisfied

Example: \( f(x_1, x_2) = x_1^3 + x_2^3 - 2x_1^2 + 3x_2^3 - 8 \).

Given an initial guess \( x^0 = [1, 1]^t \),

want to find the local minimum \( [\frac{4}{3}, 0] \) of \( f \)

\( \nabla f(x^0) = [-1, -3]^t, \; t^* = \frac{1}{3}, \; x^{(1)} = [\frac{4}{3}, 0]^t \)
Conjugate Gradient Method

**Definition:** Let \( A \in \mathbb{R}^{n\times n} \) be symmetric and positive definite. A set of vectors \( u_1, u_2, \ldots, u_n \) is said to be \( A \)-orthonormal if \( u_i^tAu_j = \delta_{ij} \).

**Theorem 1:** Let \( u_1, u_2, \ldots, u_n \) to be \( A \)-orthonormal system. Define

\[
x_i = x_{i-1} + [u_i^t(b - Ax_{i-1})]u_i \quad \text{for} \quad 1 \leq i \leq n
\]

where \( x_0 \in \mathbb{R}^{n\times n} \) is an arbitrary vector. Then \( Ax_n = b \)

---

**Conjugate Gradient Algorithm**

This algorithm solves \( Ax = b \) iteratively

Input : \( n, A, b, x_0, \varepsilon \)

\[
r_0 \leftarrow b - Ax_0
\]

\[
v_0 \leftarrow r_0
\]

for \( k = 0, 1, \ldots, n-1 \) do

if \( v_k = 0 \) return

\[
t_k \leftarrow r_k^tv_k/Av_k
\]

\[
x_{k+1} \leftarrow x_k + t_kv_k
\]

\[
r_{k+1} \leftarrow r_k - t_kAv_k
\]

if \( ||r_{k+1}||_2 < \varepsilon \) return

\[
s_k \leftarrow r_{k+1}^tv_{k+1}/r_k^tr_k
\]

\[
v_{k+1} \leftarrow r_{k+1} + s_kv_k
\]

endfor
**Theorem 2:** In the conjugate gradient algorithm, for any integer $m < n$, if $v_0, v_1, \ldots, v_m$ are all nonzero vectors, then $r_i = b - Ax_i$ for $0 \leq i \leq m$, and $r_0, r_1, \ldots, r_m$ is an orthogonal set of nonzero vectors.

**Proof:** Prove the following statements by induction on $m$.

(a) $(r_m, v_i) = 0$ for $0 \leq i < m$
(b) $(r_i, r_i) = (r_i, v_i)$ for $0 \leq i \leq m$
(c) $(v_m, Av_i) = 0$ for $0 \leq i < m$
(d) $r_i = b - Ax_i$ for $0 \leq i \leq m$
(e) $(r_m, r_i) = 0$ for $0 \leq i < m$
(f) $r_i \neq 0$ for $0 \leq i \leq m$
Simulated Annealing Procedure

To avoid the trap of a local optimum and the computation of derivative.

• Traveling Salesman Problem (TSP) which is NP-complete
• Image segmentation using contextual information.

The 10th Int’l Conference on Pattern Recognition, pp.808-814, 1990 by
R.C. Dubes, A.K. Jain, S.G. Nadabar, and C.C. Chen

The following elements must be provided.

(1) A description of system configuration.

(2) A generation of random changes into the configuration, these changes are the *options* presented to the system.

(3) An objective function $E$ (analogy of energy) whose minimization is the goal of the procedure.

(4) A control parameter $T$ (analogy of temperature) and an annealing schedule telling how it is lowered from high to low values.
Simulated Annealing for Solving Traveling Salesman Problem

(1) Configuration:

The cities are numbered $i = 1, 2, \ldots, N$. Each has coordinates $(x_i, y_i)$. A configuration is a permutation of the numbers $1, 2, \cdots, N$ interpreted as the order in which the cities are visited.

(2) Rearrangements:

(a) A section of path is removed and then replaced with the same cities running in the opposite order, or

(b) A section of path is removed and then replace in between two cities on another, randomly chosen, part of the path.

(3) Objective Function:

$$E = \sum_{i=1}^{N} [(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2]^{\frac{1}{2}}$$

(4) Annealing Schedule:

(a) An initial temperature $T_0 \gg$ maximum of possible $\triangle E$.

(b) $T_k = \frac{T_{k-1}}{m(k+1)}$

(c) For each $T_k$, we run for, say, 10N successful reconfigurations.
The term *linear programming* means finding the maximum or minimum value of a linear function of $n$ variables over a convex polyhedral set in $\mathbb{R}^n$.

**Problem:** Let $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$. Find the maximum of $c^tx$ subject to the constraints $x \in \mathbb{R}^n$, $Ax \leq b$, and $x \geq 0$.

The *objective* function is $c^tx = \sum_{i=1}^n c_i x_i$.

The *feasible* set is defined as $K = \{x \in \mathbb{R}^n : Ax \leq b, \ x \geq 0\}$

The *value* of the problem is $v = \sup \{c^tx : x \in K\}$.

- The problem is also referred to as Problem($A,b,c$)
- A linear programming problem may or may not have a solution.

With any problem ($A$, $b$, $c$), we can associate another problem ($-A^t$, $-c$, $-b$). This problem is called the dual of the original.

**Standard Form**

Maximize $c^tx$ subject to constraints $Ax=b$ and $x \geq 0$.

Here $x, c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, and $A \in \mathbb{R}^{m \times n}$.
**Theorem 1**: If \( \mathbf{x} \) is a feasible point for a linear programming problem \((A, b, c)\), and if \( \mathbf{y} \) is a feasible point for the dual problem \((-A^t, -c, -b)\), then

\[
c'\mathbf{x} \leq y^tA\mathbf{x} \leq b'y
\]

If equality occurs here, then \( \mathbf{x} \) and \( \mathbf{y} \) are solutions of their respective problems.

**Theorem 2**: If a linear programming problem and its dual both have feasible points, then both problems have solutions, and their values are the negatives of each other.

**Theorem 3**: If a linear programming problem or its dual has a solution so has the other.

**Theorem 4**: Let \( \mathbf{x} \) and \( \mathbf{y} \) be feasible points for a linear programming problem and its dual, respectively. These points are solutions of their respective problems iff \((A\mathbf{x})_i = b_i\) for each index \( i \) such that \( y_i > 0 \), and \((A\mathbf{y})_i = c_i\) for each index \( i \) such that \( x_i > 0 \).