Optimization Algorithms: Zero-Order and First-Order

Yinyu Ye

http://www.stanford.edu/~yyye

Chapters 4.2, 7, 8, 9.1-7, 12.3-6

Introduction

Optimization algorithms tend to be iterative procedures. Starting from a given point $\mathbf{x}^0$, they generate a sequence $\{\mathbf{x}^k\}$ of iterates (or trial solutions) that converge to a "solution" – or at least they are designed to be so.

Recall that scalars $\{x^k\}$ converges to 0 if and only if for all real numbers $\varepsilon > 0$ there exists a positive integer K such that

$$|x^k| < \varepsilon \quad \text{for all } k \geq K.$$

Then $\{\mathbf{x}^k\}$ converges to solution $\mathbf{x}^*$ if and only if $\{\|\mathbf{x}^k - \mathbf{x}^*\|\}$ converges to 0.

We study algorithms that produce iterates according to

- well determined rules–Deterministic Algorithm
- random selection process–Randomized Algorithm.

The rules to be followed and the procedures that can be applied depend to a large extent on the characteristics of the problem to be solved.

The Meaning of “Solution”

What is meant by a solution may differ from one algorithm to another.

In some cases, one seeks a local minimum; in some cases, one seeks a global minimum; in others, one seeks a first-order and/or second-order stationary or KKT point of some sort as in the method of steepest descent discussed below.

In fact, there are several possibilities for defining what a solution is. Once the definition is chosen, there must be a way of testing whether or not an iterate (trial solution) belongs to the set of solutions. For example, the residuals of the KKT conditions converge to zero.

Generic Algorithms for Minimization and Global Convergence Theorem

A Generic Algorithm: A point to set mapping in a subspace of R^n.

Theorem 1 *(Page 222, L&Y) Let A be an "algorithmic mapping" defined over set X, and let solution sequence $\{\mathbf{x}^k\}$, starting from a given point $\mathbf{x}^0$, be generated from current or earlier point(s):*

$$\mathbf{x}^{k+1} \in A(\mathbf{x}^k), \quad \text{or} \quad \mathbf{x}^{k+1} \in A(\mathbf{x}^k, \mathbf{x}^{k-1}, \ldots).$$

Let a solution set $S \subset X$ be given, and suppose

i) all points $\{\mathbf{x}^k\}$ are in a compact set;

ii) there is a continuous (merit) function $z(\mathbf{x})$ such that if $\mathbf{x} \notin S$, then $z(\mathbf{y}) < z(\mathbf{x})$ for all $\mathbf{y} \in A(\mathbf{x})$; otherwise, $z(\mathbf{y}) \leq z(\mathbf{x})$ for all $\mathbf{y} \in A(\mathbf{x})$;

iii) the mapping A is closed at points outside S ($\mathbf{x}^k \rightarrow \bar{\mathbf{x}} \in X$ and $A(\mathbf{x}^k) = \mathbf{y}^k \rightarrow \bar{\mathbf{y}}$ imply $\bar{\mathbf{y}} \in A(\bar{\mathbf{x}})$).

Then, the limit of any convergent subsequences of $\{\mathbf{x}^k\}$ is a solution in S.

Descent Direction Methods

In this case, merit function $z(\mathbf{x}) = f(\mathbf{x})$, that is, just the objective itself.

(A1) Test for convergence If the termination conditions are satisfied at $\mathbf{x}^k$, then it is taken (accepted) as a "solution." In practice, this may mean satisfying the desired conditions to within some tolerance. If so, stop. Otherwise, go to step (A2).

(A2) Compute a search direction, say $\mathbf{d}^k \neq \mathbf{0}$. This might be a direction in which the function value is known to decrease within the feasible region.

(A3) Compute a step-size or learning rate, say α^k such that

$$f(\mathbf{x}^k + \alpha^k \mathbf{d}^k) < f(\mathbf{x}^k).$$

This may necessitate a one-dimensional (or line) search/optimization.

(A4) Define the new iterate by setting

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k$$

and return to step (A1).

Algorithm Complexity and Speeds I

The intrinsic computational cost/time of an algorithm depends on

- number of decision variables n: cost of the inner product of two vectors, cost of solving system of linear equations
- number of constraints m: cost of the product of a matrix and a vector, cost of the product of two matrices
- number of nonzero data entries NNZ: sparse matrix/data representation
- the desired accuracy $0 \le \epsilon < 1$: the cost could be propotional to $\frac{1}{\epsilon^2},\ \frac{1}{\epsilon},\ \log(\frac{1}{\epsilon}),\ \log[\log(\frac{1}{\epsilon})], \ldots$
- problem difficulty or complexity measures such as the Lipschiz constant β, the condition number of a matrix, etc

Algorithm Complexity and Speeds II

- Finite versus infinite convergence. For some classes of optimization problems there are algorithms that obtain an exact solution—or detect the unboundedness–in a finite number of iterations.
- Polynomial-time versus exponential-time. The solution time grows, in the worst-case, as a function of problem sizes (number of variables, constraints, accuracy, etc.).
- Convergence order and rate. If there is a positive numberγ such that

$$\|\mathbf{x}^k - \mathbf{x}^*\| \leq \frac{O(1)}{k^\gamma}\|\mathbf{x}^0 - \mathbf{x}^*\|,$$

 then $\{\mathbf{x}^k\}$ converges arithmetically to $\mathbf{x}^*$ with power γ. If there exists a number $\gamma \in [0,1)$ such that

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \gamma\|\mathbf{x}^k - \mathbf{x}^*\| \quad (\Rightarrow \|\mathbf{x}^k - \mathbf{x}^*\| \leq \gamma^k\|\mathbf{x}^0 - \mathbf{x}^*\|),$$

 then $\{\mathbf{x}^k\}$ converges geometrically or linearly to $\mathbf{x}^*$ with rate γ. If there exists a number $\gamma \in [0,1)$

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \gamma\|\mathbf{x}^k - \mathbf{x}^*\|^2 \text{ after } \gamma\|\mathbf{x}^k - \mathbf{x}^*\| < 1$$

 then $\{\mathbf{x}^k\}$ converges quadratically to $\mathbf{x}^*$ (such as $\left\{(\frac{1}{2})^{2^k}\right\}$).

Algorithm Classes

Depending on information of the problem being used to create a new iterate, we have

(a) Zero-order algorithms. Popular when the gradient and Hessian information are difficult to obtain, e.g., no explicit function forms are given, functions are not differentiable, etc.

(b) First-order algorithms. Most popular now-days, suitable for large scale data optimization with low accuracy requirement, e.g., Machine Learning, Statistical Predictions...

(c) Second-order algorithms. Popular for optimization problems with high accuracy need, e.g., some scientific computing, etc.

One-Variable Optimization: Golden Section (Zero Order) Method

Assume that the one variable function $f(x)$ is Unimodel in interval $[a\ b]$, that is, for any point $x \in [a_r\ b_l]$ such that $a \leq a_r < b_l \leq b$, we have that $f(x) \leq \max\{f(a_r),\ f(b_l)\}$. How do we find x^* within an error tolerance ϵ?

0) Initialization: let $x_l = a,\ x_r = b$, and choose a constant $0 < r < 0.5$;

1) Let two other points $\hat{x}_l = x_l + r(x_r - x_l)$ and $\hat{x}_r = x_l + (1-r)(x_r - x_l)$, and evaluate their function values.

2) Update the triple points $x_r = \hat{x}_r, \hat{x}_r = \hat{x}_l, x_l = x_l$ if $f(\hat{x}_l) < f(\hat{x}_r)$; otherwise update the triple points $x_l = \hat{x}_l, \hat{x}_l = \hat{x}_r, x_r = x_r$; and return to Step 1.

In either cases, the length of the new interval after one golden section step is $(1-r)$. If we set $(1-2r)/(1-r) = r$, then only one point is new in each step and needs to be evaluated. This give $r = 0.382$ and the linear convergence rate is 0.618.

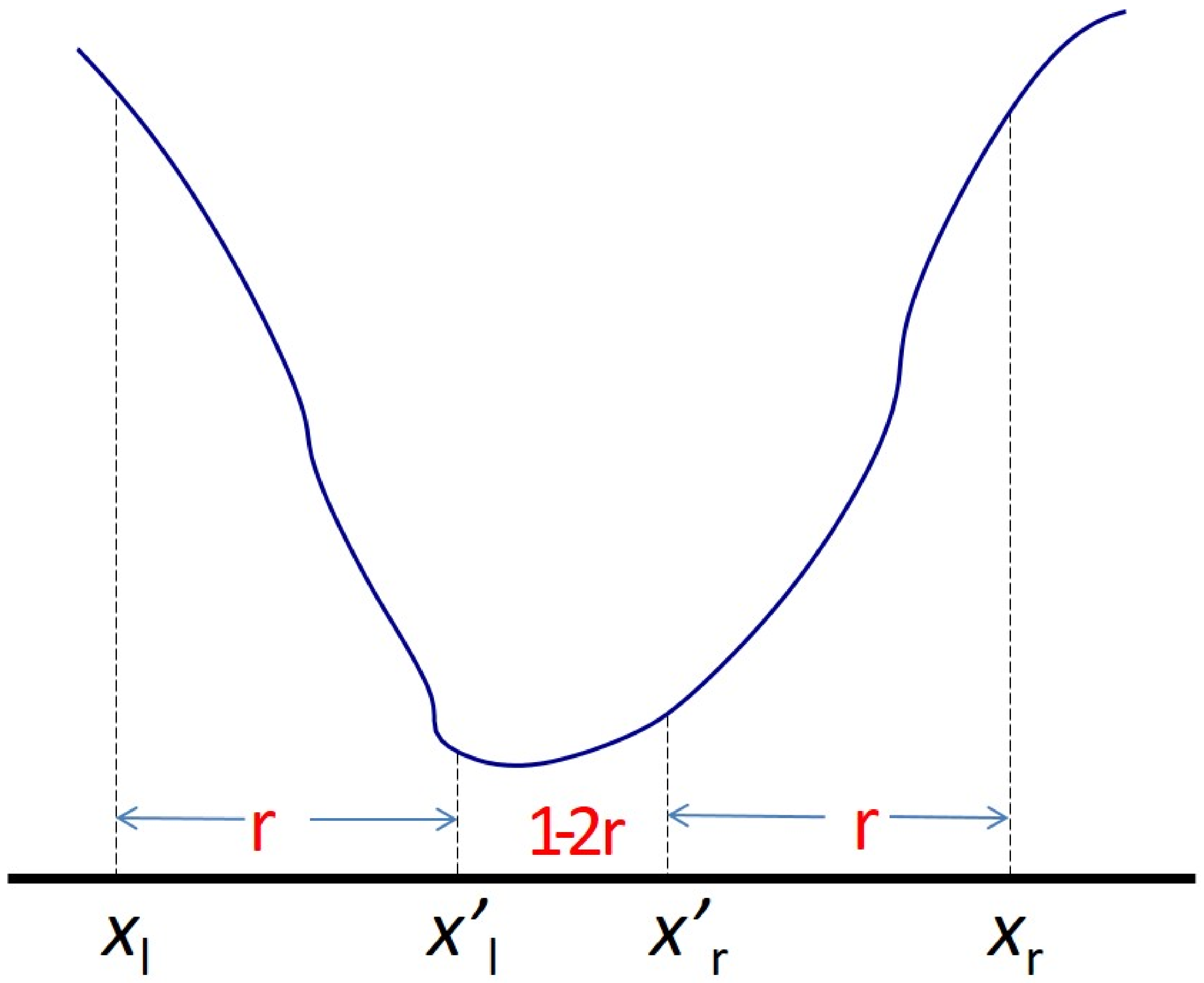

Figure 1: Illustration of Golden Section

One-Variable Optimization: Bisection (First Order) Method

For a one variable problem, an KKT point is the root of $g(x) := f'(x) = 0$.

Assume we know an interval $[a\ b]$ such that $a < b$, and $g(a)g(b) < 0$. Then we know there exists an x^*, $a < x^* < b$, such that $g(x^*) = 0$; that is, interval $[a\ b]$ contains a root of g. How do we find x within an error tolerance ϵ, that is, $|x - x^*| \le \epsilon$?

0) Initialization: let $x_l = a,\ x_r = b$.

1) Let $x_m = (x_l + x_r)/2$, and evaluate $g(x_m)$.

2) If $g(x_m) = 0$ or $x_r - x_l < \epsilon$ stop and output $x^* = x_m$. Otherwise, if $g(x_l) \cdot g(x_m) > 0$ set $x_l = x_m$; else set $x_r = x_m$; and return to Step 1.

The length of the new interval containing a root after one bisection step is $1/2$ which gives the linear convergence rate is $1/2$, and this establishes a linear convergence rate 0.5.

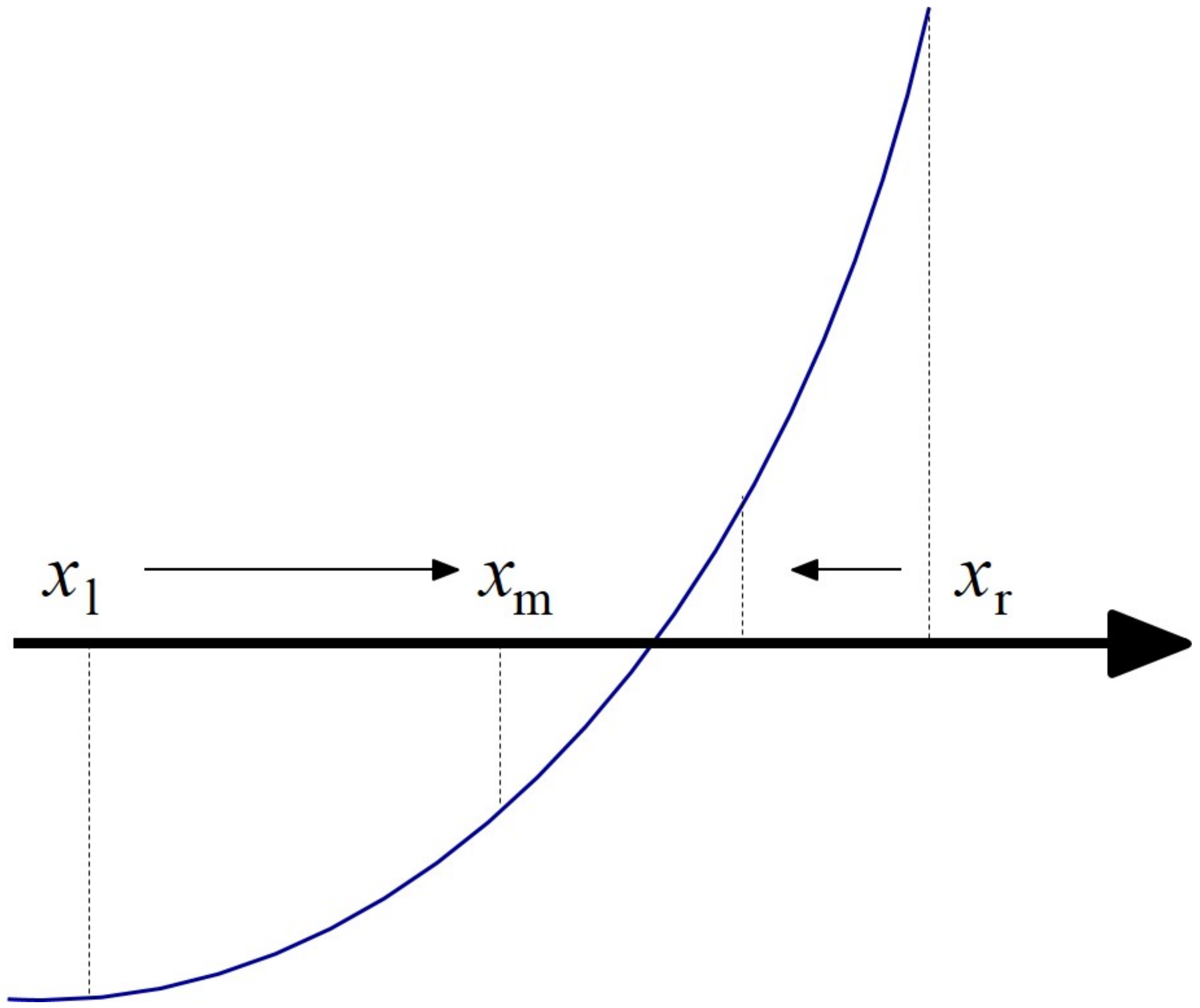

Figure 2: Illustration of Bisection

K-section in Parallel?

One-Variable Optimization: Newton's (Second Order) Method

For functions of a single real variable x, the KKT condition is $g(x) := f'(x) = 0.$ When f is twice continuously differentiable then g is once continuously differentiable, Newton's method can be a very effective way to solve such equations and hence to locate a root of g. Given a starting point x^0, Newton's method for solving the equation $g(x) = 0$ is to generate the sequence of iterates from one-point mapping:

$$x^{k+1} = x^k - \frac{g(x^k)}{g'(x^k)}.$$

The iteration is well defined provided that $g'(x^k) \neq 0$ at each step.

For strictly convex function, Newton's method has a linear convergence rate and, when the point is "close" to the root, the convergence becomes quadratic, which leads to the iterations bound of $\log[\log(\frac{1}{\epsilon})]$.

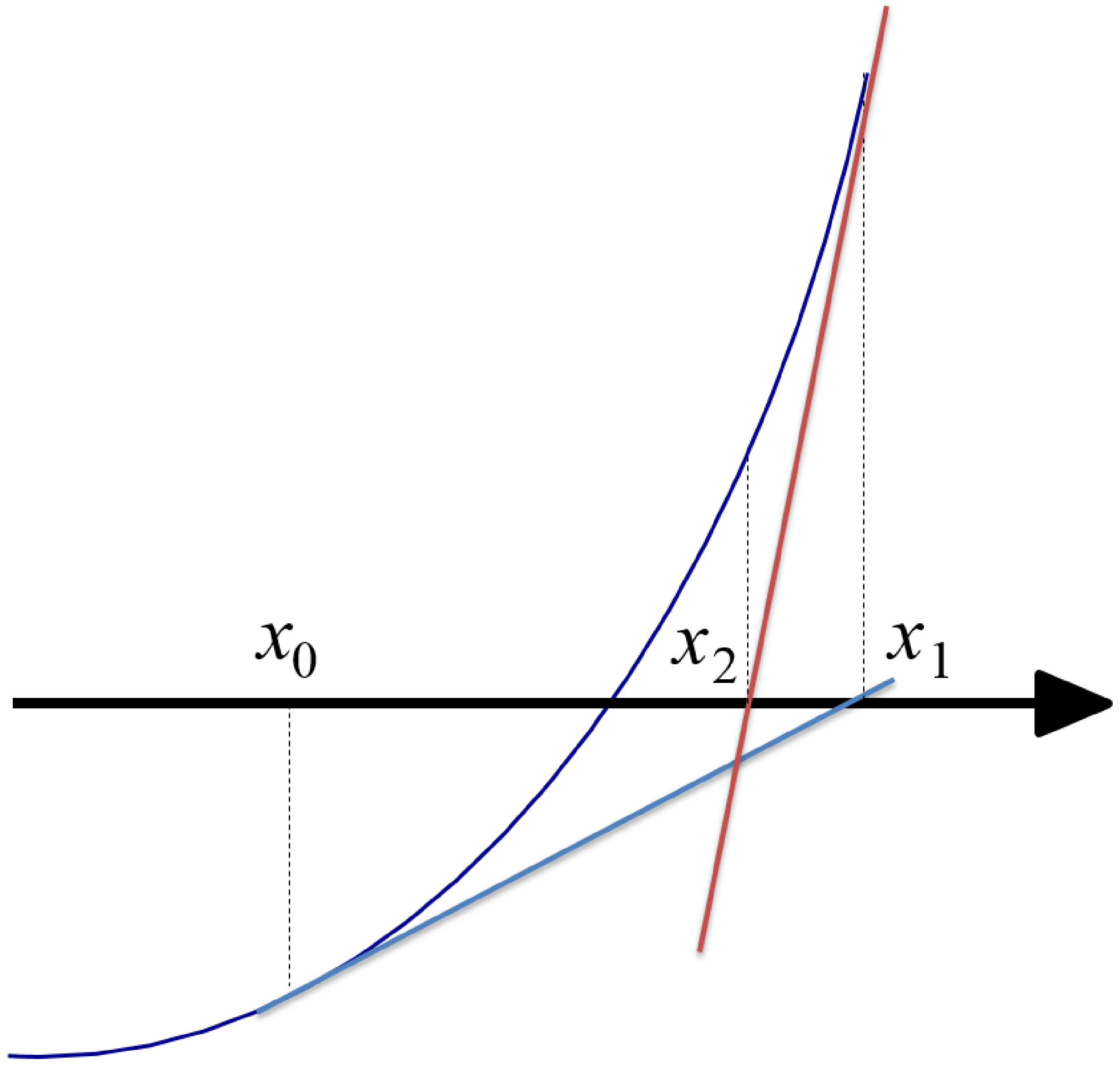

Figure 3: Illustration of Newton's Method

Hybrid of Bisection and Newton

Consider finding a root in $(0, R)$. When a containing interval with qudratic convergence becomes wider and wider at a geometric rate if the root is increased, we can construct a sequence of points:

$$\hat{x}_0 = \epsilon,\ \hat{x}_1 = (1+1/\alpha)\hat{x}_0, ...,\ \text{and } \hat{x}_j = (1+1/\alpha)\hat{x}_{j-1}, ...$$

until $\hat{x}_j = \hat{x}_J \geq R$. Obviously the total number of points, J, of these points is bounded by $O(\log(R/\epsilon))$. Also, define a sequence of intervals

$$I_j = [\hat{x}_{j-1}, \hat{x}_j] = [\hat{x}_{j-1}, (1+1/\alpha)\hat{x}_{j-1}].$$

Now if the root $\bar{x}$ of g is in any one of these intervals, say in I_j, then start at any point $\hat{x}_{j-1}$ in the interval, Newton's method generates an x with $|x - \bar{x}| \leq \epsilon$ in $O(\log\log(1/\epsilon))$ iterations.

How to identify the interval that contains $\bar{x}$? This time, we bisect the number of intervals, that is, evaluate function value at point $\hat{x}_{j_m}$ where $j_m = [J/2]$. Thus, each bisection reduces the total number of the intervals by a half. Since the total number of intervals is $O(\log(R/\epsilon))$, in at most $O(\log\log(R/\epsilon))$ bisection steps we shall locate the interval that contains $\bar{x}$. Thus the total number iterations, including both bisection and Newton methods, is $O(\log\log(R/\epsilon))$ iterations.

Here we take advantage of the global convergence property of Bisection and local quadratic convergence property of Newton, and we would see more of these features later...

Example: Spherical Constrained Nonconvex Quadratic Minimization

$$\text{min } \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}, \quad \text{s.t.} \quad \|\mathbf{x}\|^2 = (\leq)1.$$

where $Q \in S^n$ is any symmetric data matrix. If $\mathbf{c} = \mathbf{0}$ this problem becomes finding the least eigenvalue of Q.

The necessary and sufficient condition (can be proved using the SDP Rank Theorem) for $\mathbf{x}$ being a global minimizer of the problem is

$$(Q + \lambda I)\mathbf{x} = -\mathbf{c}, \; (Q + \lambda I) \succeq \mathbf{0}, \; \|\mathbf{x}\|_2^2 = 1,$$

which implies $\lambda \geq -\lambda_{min}(Q) > 0$ where $\lambda_{min}(Q)$ is the least eigenvalue of Q. This can be cast as a one-dimetnional root finding problem of scalar variable λ.

Theorem 2 *The 1-spherical constrained quadratic minimization can be computed in $O(\log\log(\|\mathbf{c}\|/\epsilon))$ iterations where each iteration solve a symmetric (positive definite) system of linear equations of n variables.*

What about 2-spherical constrained quadratic minimization, that is, quadratic minimization with 2 ellipsoidal constraints: Remains Open.

Multi-Variable Zero-Order Algorithms: the Finite-Difference Gradient or Surrogate Objective based on Sampling

$$\nabla f(\mathbf{x}^k)_j \sim \frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{e}_j) - f(\mathbf{x}^k)\right)\ \forall j$$

for a small $\delta(>0)$, and they can be estimated in parallel.

Randomized Finite-Difference Gradient Randomly select a block of variables $B \subset of\{1, 2, ..., n\}$ and approximate the gradient vector by

$$\nabla f(\mathbf{x}^k) \sim \frac{n}{|B|}\sum_{j\in B}[\frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{e}_j) - f(\mathbf{x}^k)\right)]\mathbf{e}_j.$$

Randomly generate n_k i.i.d. Gaussian vectors $\mathbf{u}_i,\ i = 1, ..., n_k$ and and approximate the gradient vector by

$$\nabla f(\mathbf{x}^k) \sim \frac{n}{n_k}\sum_{i=1}^{n_k}[\frac{1}{\delta}\left(f(\mathbf{x}^k + \delta \mathbf{u}_i) - f(\mathbf{x}^k)\right)]\mathbf{u}_i.$$

Surrogate objective based on Sampling sample a number of point ans use their function values to build a (local) quadratic surrogate-objective function to optimize.

(SOLNP+, ongoing work)

First-Order Algorithms: the Steepest Descent Method (SDM)

Let f be a differentiable function and assume we can compute gradient (column) vector ∇f . We want to solve the unconstrained minimization problem

$$\min_{\mathbf{x} \in R^n} f(\mathbf{x}).$$

In the absence of further information, we seek a first-order KKT or stationary point of f, that is, a point $\mathbf{x}^*$ at which $\nabla f(\mathbf{x}^*) = \mathbf{0}$. Here we choose direction vector $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$ as the search direction at $\mathbf{x}^k$, which is the direction of steepest descent.

The number $\alpha^k \geq 0$, called step-size or learning rate, is chosen "appropriately" such as one-point mapping:

$$\alpha^k \in \arg\min f(\mathbf{x}^k - \alpha \nabla f(\mathbf{x}^k)).$$

Then the new iterate is defined as $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla f(\mathbf{x}^k)$.

In some implementations, step-size α^k is fixed through out the process – independent of iteration count k

SDM Example: Unconstrained Convex Quadratic Optimization

Let $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x}$ where $Q \in R^{n\times n}$ is symmetric and positive definite. This implies that the eigenvalues of Q are all positive. The unique minimum $\mathbf{x}^*$ of $f(\mathbf{x})$ exists and is given by the solution of the system of linear equations

$$\nabla f(\mathbf{x})^T = Q\mathbf{x} + \mathbf{c} = \mathbf{0},$$

or equivalently

$$Q\mathbf{x} = -\mathbf{c}.$$

The iterative scheme becomes, from $\mathbf{d}^k = -(Q\mathbf{x}^k + \mathbf{c})$,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k = \mathbf{x}^k - \alpha^k(Q\mathbf{x}^k + \mathbf{c});$$

where

$$\alpha^k = \frac{\|\mathbf{d}^k\|^2}{(\mathbf{d}^k)^\top Q\mathbf{d}^k}.$$

Note that minimizing a strictly convex quadratic function is equivalent to solve a system of equation with a positive definite matrix on the left. The iterative method maybe preferable if the system only needs to be solved approximately and it is very large.

(linesteepestqp.m of Chapter 8)

Iterate Convergence of the Steepest Descent Method

The following theorem gives some conditions under which the steepest descent method will generate a sequence of iterates that converge .

Theorem 3 *Let $f : R^n \rightarrow R$ be given. For some given point $\mathbf{x}^0 \in R^n$, let the level set*

$$X^0 = \{\mathbf{x} \in R^n : f(\mathbf{x}) \leq f(\mathbf{x}^0)\}$$

be bounded. Assume further that f is continuously differentiable on the convex hull of X^0. Let $\{\mathbf{x}^k\}$ be the sequence of points generated by the steepest descent method initiated at $\mathbf{x}^0$. Then every accumulation point of $\{\mathbf{x}^k\}$ is a stationary point of f.

Remark According to this theorem, the steepest descent method initiated at any point of the level set X^0 will converge to a stationary point of f, which property is called global convergence.

This proof can be viewed as a special form of Theorem 1: the SDM algorithm mapping is closed and the objective function is strictly decreasing if not optimal yet.

Convergence Speed of the SDM for Strongly Convex QP

The convergence rate of the steepest descent method applied to convex quadratic functions is known to be linear. Suppose Q is a symmetric positive definite matrix of order n and let its eigenvalues be $0 < \lambda_1 \leq \cdots \leq \lambda_n$. Obviously, the global minimizer of the quadratic form $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x}$ is at the origin.

It can be shown that when the steepest descent method is started from any nonzero point $\mathbf{x}^0 \in R^n$, there will exist constants c_1 and c_2 such that (page 235, L&Y)

$$0 < c_1 \leq \frac{f(\mathbf{x}^{k+1})}{f(\mathbf{x}^k)} \leq c_2 \leq \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}\right)^2 < 1, \; k = 0, 1, \ldots \; .$$

λ_n/λ_1 is called condition number of the Hessian matrix. Intuitively, the slow rate of linear convergence of the steepest descent method can be attributed the fact that the successive search directions are perpendicular.

Consider an arbitrary iterate $\mathbf{x}^k$. At this point we have the search direction $\mathbf{d}^k = -\nabla f(\mathbf{x}^k)$. To find the next iterate $\mathbf{x}^{k+1}$ we minimize $f(\mathbf{x}^k - \alpha\nabla f(\mathbf{x}^k))$ with respect to $\alpha \geq 0$. At the minimum α^k, the derivative of this function will equal zero. Thus, we obtain $\nabla f(\mathbf{x}^{k+1})^T \nabla f(\mathbf{x}^k) = 0$.

Convergence Speed of the SDM for Minimizing Lipschitz Functions

Let $f(\mathbf{x})$ be differentiable every where and satisfy the (first-order) β-Lipschitz condition, that is, for any two points $\mathbf{x}$ and $\mathbf{y}$

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le \beta\|\mathbf{x}-\mathbf{y}\| \tag{1}$$

for a positive real constant β. Then, we have

$$f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \le \frac{\beta}{2}\|\mathbf{x}-\mathbf{y}\|^2 \tag{2}$$

so that at the kth step of SDM, we have

$$f(\mathbf{x}) - f(\mathbf{x}^k) \le \nabla f(\mathbf{x}^k)^T(\mathbf{x}-\mathbf{x}^k) + \frac{\beta}{2}\|\mathbf{x}-\mathbf{x}^k\|^2.$$

Let us minimize the quadratic function and then it has a close form: $\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{\beta}\nabla f(\mathbf{x}^k)$ which is the SDM with the fixed step-size $\frac{1}{\beta}$. One can prove

Theorem 4 *(Error Convergence Estimate Theorem) Let the objective function* $p^* = \inf\ f(\mathbf{x})$ *be finite and let us stop the SDM as soon as* $\|\nabla f(\mathbf{x}^k)\| \le \epsilon$ *for a given tolerance* $\epsilon \in (0\ 1)$. *Then the SDM terminates in* $\frac{2\beta(f(\mathbf{x}^0)-p^*)}{\epsilon^2}$ *steps.*

The convergence speed can be improved to $\frac{\beta(f(\mathbf{x}^0)-p^*)}{\epsilon}$ steps if the quadratic function is convex.

(steepestqp.m of Chapter 8)

The Barzilai and Borwein Method: Two-Point Algorithm Mapping

There is a steepest descent method (Barzilai and Borwein 88) that chooses the step-size based on the two past points/solutions as follows:

$$\Delta_x^k = \mathbf{x}^k - \mathbf{x}^{k-1} \quad \text{and} \quad \Delta_g^k = \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1}), \tag{3}$$

$$\alpha^k = \frac{(\Delta_x^k)^T \Delta_g^k}{(\Delta_g^k)^T \Delta_g^k} \quad \text{or} \quad \alpha^k = \frac{(\Delta_x^k)^T \Delta_x^k}{(\Delta_x^k)^T \Delta_g^k},$$

Then

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla f(\mathbf{x}^k). \tag{4}$$

For convex quadratic minimization with Hessian Q, $\Delta_g^k = Q\Delta_x^k$, the two step size formula become

$$\alpha^k = \frac{(\Delta_x^k)^T Q \Delta_x^k}{(\Delta_x^k)^T Q^2 \Delta_x^k} \quad \text{or} \quad \alpha^k = \frac{(\Delta_x^k)^T \Delta_x^k}{(\Delta_x^k)^T Q \Delta_x^k}$$

and it is between the reciprocals of the largest and smallest non-zero eigenvalues of Q (Rayleigh quotient). (BBsteepestqp.m of Chapter 8)

Another Two-Point Algorithm Mapping: The QP Heavy-Ball Method (Polyak 64)

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{4}{(\sqrt{\lambda_n} + \sqrt{\lambda_1})^2} \nabla f(\mathbf{x}^k) + \left(\frac{\sqrt{\lambda_n} - \sqrt{\lambda_1}}{\sqrt{\lambda_n} + \sqrt{\lambda_1}} \right) (\mathbf{x}^k - \mathbf{x}^{k-1}).$$

where the convergence rate can be improved to

$$\left(\frac{\sqrt{\lambda_n} - \sqrt{\lambda_1}}{\sqrt{\lambda_n} + \sqrt{\lambda_1}} \right)^2 .$$

This is also called the Parallel-Tangent or Conjugate Direction method, where the second direction-term in the formula is nowadays called “acceleration” or “momentum” direction (HBsteepestqp.m of Chapter 8).

More generally, let momentum direction $\mathbf{d}^k = (\mathbf{x}^k - \mathbf{x}^{k-1})$ and

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_1 \nabla f(\mathbf{x}^k) + \alpha_2 \mathbf{d}^k = \mathbf{x}^k + \mathbf{d}(\alpha),$$

where the pair of step-sizes $\alpha = (\alpha_1; \alpha_2)$ can be chosen from

$$\min_{(\alpha)} \; \nabla f(\mathbf{x}^k)\mathbf{d}(\alpha) + \frac{1}{2}\mathbf{d}(\alpha)\nabla^2 f(\mathbf{x}^k)\mathbf{d}(\alpha),$$

where $\mathbf{x}^1$ can be computed from the SDM step.

The momentum dirction can be viwed as an aggregation of all past gradients.

Dimension-Reduced Second-Order Method: Implicit Three-Point Algorithm Mapping

Let $\mathbf{d}^k = \mathbf{x}^k - \mathbf{x}^{k-1}$, $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$ and $H^k = \nabla^2 f(\mathbf{x}^k)$, then the step-sizes can be explicitly computed from

$$\begin{pmatrix} (\mathbf{g}^k)^T H^k \mathbf{g}^k & -(\mathbf{d}^k)^T H^k \mathbf{g}^k \\ -(\mathbf{d}^k)^T H^k \mathbf{g}^k & (\mathbf{d}^k)^T H^k \mathbf{d}^k \end{pmatrix} \begin{pmatrix} \alpha^g \\ \alpha^m \end{pmatrix} = \begin{pmatrix} \|\mathbf{g}^k\|^2 \\ -(\mathbf{g}^k)^T \mathbf{d}^k \end{pmatrix}.$$

If the Hessian $\nabla^2 f(\mathbf{x}^k)$ is not available, one can approximate

$$H^k \mathbf{g}^k \sim \nabla(\mathbf{x}^k + \mathbf{g}^k) - \mathbf{g}^k \quad \text{and} \quad H^k \mathbf{d}^k \sim \nabla(\mathbf{x}^k + \mathbf{d}^k) - \mathbf{g}^k \sim -(\mathbf{g}^{k-1} - \mathbf{g}^k);$$

or for some small $\epsilon > 0$:

$$H^k \mathbf{g}^k \sim \frac{1}{\epsilon}(\nabla(\mathbf{x}^k + \epsilon \mathbf{g}^k) - \mathbf{g}^k) \quad \text{and} \quad H^k \mathbf{d}^k \sim \frac{1}{\epsilon}(\nabla(\mathbf{x}^k + \epsilon \mathbf{d}^k) - \mathbf{g}^k).$$

For convex quadratic minimization, the method becomes the Conjugate-Gradient or Parallel-Tangent method. (DRSOMqp.m of Chapter 8)

The Accelerated Steepest Descent Method (ASDM)

There is an accelerated steepest descent method (Nesterov 83) that works as follows:

$$\lambda^0 = 0,\ \lambda^{k+1} = \frac{1+\sqrt{1+4(\lambda^k)^2}}{2},\ \alpha^k = \frac{1-\lambda^k}{\lambda^{k+1}}, \tag{5}$$

$$\tilde{\mathbf{x}}^{k+1} = \mathbf{x}^k - \frac{1}{\beta}\nabla f(\mathbf{x}^k),\ \mathbf{x}^{k+1} = (1-\alpha^k)\tilde{\mathbf{x}}^{k+1} + \alpha^k\tilde{\mathbf{x}}^k. \tag{6}$$

Note that $(\lambda^k)^2 = \lambda^{k+1}(\lambda^{k+1}-1)$, $\lambda^k > k/2$ and $\alpha^k \leq 0$.

One can prove:

Theorem 5

$$f(\tilde{\mathbf{x}}^{k+1}) - f(\mathbf{x}^*) \leq \frac{2\beta}{k^2}\|\mathbf{x}^0 - \mathbf{x}^*\|^2,\ \forall k \geq 1.$$

(accelsteepestqp.m of Chapter 8)

Applications: Federated Regression Learning of using FOM and DRSOM, Chapter 8.

First-Order Algorithms for Conic Constrained Optimization (CCO)

Consider the conic nonlinear optimization problem: min $f(\mathbf{x})$ s.t. $\mathbf{x} \in K$.

- Nonnegative Linear Regression: given data $A \in R^{m\times n}$ and $\mathbf{b} \in R^m$

$$\text{min } f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2 \text{ s.t. } \mathbf{x} \geq \mathbf{0}; \quad \text{where } \nabla f(\mathbf{x}) = A^T(A\mathbf{x} - \mathbf{b}).$$

- Semidefinite Linear Regression: given data $A_i \in S^n$ for $i = 1, ..., m$ and $\mathbf{b} \in R^m$

$$\text{min } f(X) = \frac{1}{2}\|\mathcal{A}X - \mathbf{b}\|^2 \text{ s.t. } X \succeq \mathbf{0}; \quad \text{where } \nabla f(X) = \mathcal{A}^T(\mathcal{A}X - \mathbf{b}).$$

$$\mathcal{A}X = \begin{pmatrix} A_1 \bullet X \\ \dots \\ A_m \bullet X \end{pmatrix} \quad \text{and} \quad \mathcal{A}^T\mathbf{y} = \sum_{i=1} y_i A_i.$$

Suppose we start from a feasible solution $\mathbf{x}^0$ or X^0.

SDM Followed by the Conic-Region-Projection

- $\hat{\mathbf{x}}^{k+1} = \mathbf{x}^k - \frac{1}{\beta}\nabla f(\mathbf{x}^k)$
- $\mathbf{x}^{k+1} = \text{Proj}_K(\hat{\mathbf{x}}^{k+1})$: Solve $\min_{\mathbf{x}\in K} \ \|\mathbf{x} - \hat{\mathbf{x}}^{k+1}\|^2$.

For examples:

- if $K = \{\mathbf{x} : \ \mathbf{x} \geq \mathbf{0}\}$, then

$$\mathbf{x}^{k+1} = \text{Proj}_K(\hat{\mathbf{x}}^{k+1}) = \max\{\mathbf{0}, \ \hat{\mathbf{x}}^{k+1}\}.$$

- If $K = \{X : \ X \succeq \mathbf{0}\}$, then factorize $\hat{X}^{k+1} = \sum_{j=1}^{n} \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ and let

$$X^{k+1} = \text{Proj}_K(\hat{X}^{k+1}) = \sum_{j:\lambda_j>0} \lambda_j \mathbf{v}_j \mathbf{v}_j^T.$$

(The drawback is that the total eigenvalue-factorization may be costly...)

Does the method converge? What is the convergence speed?

(steepestnnqp.m of Chapter 8)

SDM Followed by the Convex-Region-Projection

Consider the convex-region-constrained nonlinear optimization problem: $\min \ f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b}$. that is $K = \{\mathbf{x} : \ A\mathbf{x} = \mathbf{b}\}$.

The projection method becomes, starting from a feasible solution $\mathbf{x}^0$ and let direction

$$\mathbf{d}^k = -(I - A^T(AA^T)^{-1}A)\nabla f(\mathbf{x}^k)$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k \mathbf{d}^k; \tag{7}$$

where the stepsize can be chosen from line-search or again simply let

$$\alpha^k = \frac{1}{\beta}$$

and β is the (global) Lipschitz constant.

Does the method converge? What is the convergence speed? See more details in HW3.

SDM Followed by the Nonconvex-Region-Projection

- $K \subset R^n$ whose support size is no more than $d(< n)$: $\mathbf{x} = \text{Proj}_K(\hat{\mathbf{x}})$ contains the largest d absolute entries of $\hat{\mathbf{x}}$ and set the rest of them to zeros.
- $K \subset R^n_+$ and its support size is no more than $d(< n)$: $\mathbf{x} = \text{Proj}_K(\hat{\mathbf{x}})$ contains the largest no more than d positive entries of $\hat{\mathbf{x}}$ and set the rest of them to zeros.
- $K \subset S^n$ whose rank is no more than $d(< n)$: factorize $\hat{X} = \sum_{j=1}^n \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ with $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$ then $\text{Proj}_K(\hat{X}) = \sum_{j=1}^d \lambda_j \mathbf{v}_j \mathbf{v}_j^T$.
- $K \subset S^n_+$ whose rank is no more than $d(< n)$: factorize $\hat{X} = \sum_{j=1}^n \lambda_j \mathbf{v}_j \mathbf{v}_j^T$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$ then $\text{Proj}_K(\hat{X}) = \sum_{j=1}^d \max\{0, \lambda_j\} \mathbf{v}_j \mathbf{v}_j^T$.

Does the method converge? What is the convergence speed? What if $f(\cdot)$ is not a convex function?

Multiplicative-Update I: "Mirror" SDM for CCO

At the kth iterate with $\mathbf{x}^k > \mathbf{0}$:

$$\mathbf{x}^{k+1} = \mathbf{x}^k .* \exp(-\frac{1}{\beta}\nabla f(\mathbf{x}^k))$$

Note that $\mathbf{x}^{k+1}$ remains positive in the updating process.

The classical Projected SDM update can be viewed as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \ \nabla f(\mathbf{x}^k)^T\mathbf{x} + \frac{\beta}{2}\|\mathbf{x}-\mathbf{x}^k\|^2.$$

One can choose any strongly convex function $h(\cdot)$ and define

$$\mathcal{D}_h(\mathbf{x},\mathbf{y}) = h(\mathbf{x}) - h(\mathbf{y}) - \nabla h(\mathbf{y})^T(\mathbf{x}-\mathbf{y})$$

and define the update as

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\geq\mathbf{0}} \ \nabla f(\mathbf{x}^k)^T\mathbf{x} + \beta\mathcal{D}_h(\mathbf{x},\mathbf{x}^k).$$

The update above is the result of choosing (negative) entropy function $h(\mathbf{x}) = \sum_j x_j \log(x_j)$.

(mirrorsteepestnnqp.m of Chapter 8)

Multiplicative-Update II: Affine Scaling SDM for CCO

At the kth iterate with $\mathbf{x}^k > \mathbf{0}$, let D^k be a diagonal matrix such that

$$D^k_{jj} = x^k_j, \ \forall j$$

and

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x} \geq \mathbf{0}} \ \nabla f(\mathbf{x}^k)^T \mathbf{x} + \frac{\beta}{2} \|(D^k)^{-1}(\mathbf{x} - \mathbf{x}^k)\|^2,$$

or

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k (D^k)^2 \nabla f(\mathbf{x}^k) = \mathbf{x}^k . * (\mathbf{e} - \alpha_k \nabla f(\mathbf{x}^k) . * \mathbf{x}^k)$$

where variable step-sizes can be

$$\alpha^k = \min\{ \frac{1}{\beta \max(\mathbf{x}^k)^2}, \ \frac{1}{2\|\mathbf{x}^k . * \nabla f(\mathbf{x}^k)\|_\infty} \}.$$

Is $\mathbf{x}^k > \mathbf{0}, \ \forall k$? Does it converge? What is the convergence speed? See more details in HW3.

Geometric Interpretation: inscribed ball vs inscribed ellipsoid.

Reduced Gradient Method – the Simplex Algorithm for LP

$$\text{LP:} \quad \min \quad \mathbf{c}^T\mathbf{x} \quad \text{s.t. } A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0},$$

where $A \in R^{m \times n}$ has a full row rank m.

Theorem 6 *(The Fundamental Theorem of LP in Algebraic form) Given (LP) and (LD) where A has full row rank m,*

i) *if there is a feasible solution, there is a basic feasible solution (Carathéodory's theorem) that is a corner point of the feasible region;*

ii) *if there is an optimal solution, there is an optimal basic solution.*

High-Level Idea:

1. Initialization Start at a BSF (corner point) of the feasible polyhedron and transform it to the "origin".
2. Test for Optimality. Compute the reduced gradient vector at the corner. If no descent and feasible direction can be found, stop and claim optimality at the current corner point; otherwise, select a new corner point and go to Step 2.

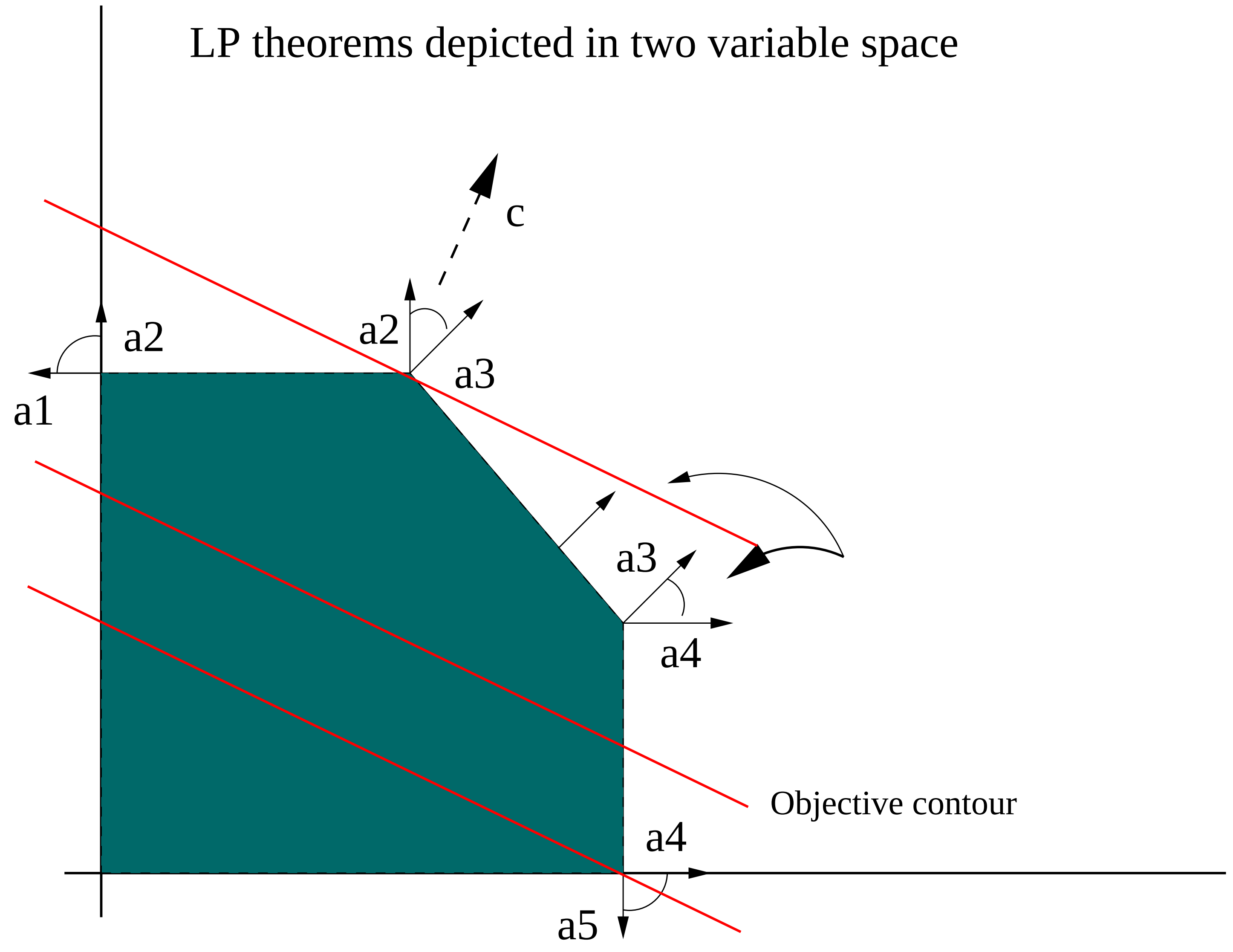

Figure 4: The LP Simplex Method

The Frank-Wolf Algorithm: SLP

$$\text{P:} \quad \min \quad f(\mathbf{x}) \quad \text{s.t. } A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0},$$

where $A \in R^{m \times n}$ has a full row rank m.

Start with a feasible solution $\mathbf{x}^0$, and at the kth iterate do:

- Solve the LP problem

$$\min \quad \nabla f(\mathbf{x}^k)^T \mathbf{x} \quad \text{s.t. } A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$$

  and let $\tilde{\mathbf{x}}^{k+1}$ be an optimal solution.

- Choose a step-size $0 < \alpha^k \leq 1$ and let

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha^k (\tilde{\mathbf{x}}^{k+1} - \mathbf{x}^k).$$

This is also called sequential linear programming (SLP) method.

First-Order Method for MDP (MGP): Value-Iteration of Fixed-Point Iteration

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, ith entry for ith state, of a given policy. The MDP problem entails choosing the optimal value vector $\mathbf{y}^*$ which is a fixed-point of:

$$y_i^* = \text{min(max)}_{j\in\mathcal{A}_i}\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*\}, \ \forall i \in I^\cdot,$$

The Value-Iteration (VI) Method is, starting from any $\mathbf{y}^0$, the iterative mapping:

$$y_i^{k+1} = A(\mathbf{y}^k)_j = \text{min(max)}_{j\in\mathcal{A}_i}\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\}, \ \forall i \in I^\cdot.$$

For MDP, if the initial $\mathbf{y}^0$ is strictly feasible for state i, that is, $y_i^0 < c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0, \ \forall j \in \mathcal{A}_i$, then y_i^k would be increasing in the VI iteration for all i and k.

On the other hand, if any of the inequalities is violated, then we have to decrease y_i^1 at least to

$$\min_{j\in\mathcal{A}_i}\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0\}$$

.

Convergence of Value-Iteration for MDP

Theorem 7 *Let the VI algorithm mapping be* $A(\mathbf{v})_i = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{v}, \ \forall i\}$. *Then, for any two value vectors* $\mathbf{u} \in R^m$ *and* $\mathbf{v} \in R^m$ *and every state* i*:*

$$|A(\mathbf{u})_i - A(\mathbf{v})_i| \leq \gamma \|\mathbf{u} - \mathbf{v}\|_\infty, \ \textit{which implies} \ \|A(\mathbf{u})_i - A(\mathbf{v})_i\|_\infty \leq \gamma \|\mathbf{u} - \mathbf{v}\|_\infty$$

Let j_u and j_v be the two arg min actions for value vectors $\mathbf{u}$ and $\mathbf{v}$, respectively. Assume that $A(\mathbf{u})_i - A(\mathbf{v})_i \geq 0$ where the other case can be proved similarly.

$$\begin{aligned} 0 \leq A(\mathbf{u})_i - A(\mathbf{v})_i &= (c_{j_u} + \gamma \mathbf{p}_{j_u}^T \mathbf{u}) - (c_{j_v} + \gamma \mathbf{p}_{j_v}^T \mathbf{v}) \\ &\leq (c_{j_v} + \gamma \mathbf{p}_{j_v}^T \mathbf{u}) - (c_{j_v} + \gamma \mathbf{p}_{j_v}^T \mathbf{v}) \\ &= \gamma \mathbf{p}_{j_v}^T (\mathbf{u} - \mathbf{v}) \leq \gamma \|\mathbf{u} - \mathbf{v}\|_\infty . \end{aligned}$$

where the first inequality is from that j_u is the arg min action for value vector $\mathbf{u}$, and the last inequality follows from the fact that the elements in $\mathbf{p}_{j_v}$ are non-negative and sum-up to 1.

Value-Iteration for MDP II: Other Strategies

- One can choose i at random to update randomly in each iteration, either with or without replacement.
- Aggregate states if they have very similar cost-to-go values: State-Trimming.
- State-values are updated in a unsynchronized manner: a state is updated based on the current state-values of its neighborhood states, and it can be synchronized or unsynchronized.

Many research issues are listed in an optional Project.

First-Order Method for Nonlinear Constrained Optimization I

We consider the general constrained optimization:

$$
(\text{GCO}) \qquad \begin{array}{lll} \text{min} & f(\mathbf{x}) & \\ \text{s.t.} & c_i(\mathbf{x}) & = 0, \; i \in \mathcal{E}, \\ & c_i(\mathbf{x}) & \geq 0, \; i \in \mathcal{I}. \end{array}
$$

- Convert it to an unconstrained problem by adding the Penalty or Barrier Term to the objective:

$$
\text{min } \; f(\mathbf{x}) + \lambda \sum_{i \in \mathcal{E}} |c_i(\mathbf{x})| - \gamma \sum_{i \in \mathcal{I}} c_i(\mathbf{x})^{-} \quad \text{or} \quad \text{min } \; f(\mathbf{x}) + \lambda \sum_{i \in \mathcal{E}} |c_i(\mathbf{x})| - \mu \sum_{i \in \mathcal{I}} \log(c_i(\mathbf{x}))
$$

  where λ and γ are sufficiently large (μ is sufficiently small for the Barrier function). One can adjust λ and μ dynamically.

- Descent-First and Feasible-Second: apply the SLP strategy to the first-order Taylor expansion to compute a solution feasible for the linearized constraints, then project it onto the nonlinear-constrained feasible region.

Block Coordinate Descent Method for Unconstrained Optimization I

$$\min_{\mathbf{x}\in R^N} \quad f(\mathbf{x}) = f((\mathbf{x}_1;\ \mathbf{x}_2,\ \ldots;\ \mathbf{x}_n)), \quad \text{where } \mathbf{x} = (\mathbf{x}_1;\ \mathbf{x}_2;\ \ldots;\ \mathbf{x}_n).$$

For presentation simplicity, we let each $\mathbf{x}_j$ be a scalar variable so that $N = n$.

Let $f(\mathbf{x})$ be differentiable every where and satisfy the (first-order) β-Coordinate Lipschitz condition, that is, for any two vectors $\mathbf{x}$ and $\mathbf{d}$

$$\|\nabla_j f(\mathbf{x} + \mathbf{e}_j .* \mathbf{d}) - \nabla_j f(\mathbf{x})\| \le \beta_j \|\mathbf{e}_j .* \mathbf{d}\| \tag{8}$$

where $\mathbf{e}_j$ is the unit vector that $e_j = 1$ and zero everywhere else, and $.*$ is the component-wise product.

Cyclic Block Coordinate Descent (CBCD) Method (Gauss-Seidel) (CyclicBCDlls.m of Chapter 8)

$$\mathbf{x}_1 \longleftarrow \text{arg min}_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots, \mathbf{x}_n),$$
$$\vdots$$
$$\mathbf{x}_n \longleftarrow \text{arg min}_{\mathbf{x}_n} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Aitken Double Sweep Method:

$$
\begin{aligned}
\mathbf{x}_1 &\longleftarrow \arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots, \mathbf{x}_n), \\
&\vdots \\
\mathbf{x}_n &\longleftarrow \arg\min_{\mathbf{x}_n} f(\mathbf{x}_1, \ldots, \mathbf{x}_n), \\
\mathbf{x}_{n-1} &\longleftarrow \arg\min_{\mathbf{x}_{n-1}} f(\mathbf{x}_1, \ldots, \mathbf{x}_n), \\
&\vdots \\
\mathbf{x}_1 &\longleftarrow \arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).
\end{aligned}
$$

Gauss-Southwell Method:

- Compute the gradient vector $\nabla f(\mathbf{x})$ and let $i^* = \arg\max\{|\nabla f(\mathbf{x})_j|\}$.
- $$\mathbf{x}_{i^*} \longleftarrow \arg\min_{\mathbf{x}_i} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Block Coordinate Descent Method for Unconstrained Optimization II

Randomly-Permuted Cyclic Block Coordinate Descent (RCBCD) Method

- Draw a random permutation $\sigma = \{\sigma(1), \ldots, \sigma(n)\}$ of $\{1, \ldots, n\}$;
-
$$\mathbf{x}_{\sigma(1)} \longleftarrow \text{arg min}_{\mathbf{x}_{\sigma(1)}} f(\mathbf{x}_1, \ldots, \mathbf{x}_n),$$
$$\vdots$$
$$\mathbf{x}_{\sigma(n)} \longleftarrow \text{arg min}_{\mathbf{x}_{\sigma(n)}} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Randomized Block Coordinate Descent (RBCD) Method (RABCDlls.m of Chapter 8)

- Randomly choose $i^* \in \{1, 2, ..., n\}$.
-
$$\mathbf{x}_{i^*} \longleftarrow \text{arg min}_{\mathbf{x}_{i^*}} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

Convergence Analyses of the BCD Methods

The following theorem gives some conditions under which the deterministic BCD method will generate a sequence of iterates that converge.

Theorem 8 *Let $f : R^n \rightarrow R$ be given. For some given point $x^0 \in R^n$, let the level set*

$$X^0 = \{\mathbf{x} \in R^n : f(\mathbf{x}) \leq f(\mathbf{x}^0)\}$$

be bounded. Assume further that f is continuously differentiable on the convex hull of X^0. Let $\{\mathbf{x}^k\}$ be the sequence of points generated by the Cyclic Block Coordinate Descent Method initiated at $\mathbf{x}^0$. Then every accumulation point of $\{\mathbf{x}^k\}$ is a stationary point of f.

For strictly convex quadratic minimization with Hessian Q, e.g., the linear convergence rate of Gauss-Southwell is

$$\left(1 - \frac{\lambda_{min}(Q)}{\lambda_{max}(Q)(n-1)}\right)^{n-1} \geq 1 - \frac{\lambda_{min}(Q)}{\lambda_{max}(Q)} \geq \left(\frac{\lambda_{max}(Q) - \lambda_{min}(Q)}{\lambda_{max}(Q) + \lambda_{min}(Q)}\right)^2 .$$

Worst-Case Convergence Comparison of BCDs

There is a convex quadratic minimization problem of dimension n:

$$\text{min} \quad \mathbf{x}^T Q\mathbf{x}, \quad \text{where for } \gamma \in (0,\ 1)$$

$$Q = \begin{pmatrix} 1 & \gamma & \ldots & \gamma \\ \gamma & 1 & \ldots & \gamma \\ \ldots & \ldots & \ldots & \ldots \\ \gamma & \gamma & \ldots & 1 \end{pmatrix}.$$

- CBCD is $\frac{n}{2\pi^2}$ times slower than SDM;
- CBCD is $\frac{n^2}{2\pi^2}$ times slower than RBCD (each iteration consists of n random selections);
- CBCD is $\frac{n(n+1)}{2\pi^2}$ times slower than RCBCD;

Randomization makes a difference.

Randomized Block Coordinate Gradient Descent Method

At the kth Iteration of RBCGD:

- Randomly choose $i^k \in \{1, 2, ..., n\}$.
- $$\mathbf{x}_{i^k}^{k+1} = \mathbf{x}_{i^k}^k - \frac{1}{\beta_{i^k}} \nabla_{i^k} f(\mathbf{x}^k),$$
  $$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k, \ \forall i \neq i^k.$$

Theorem 9 *(Expected Error Convergence Estimate Theorem) Let the objective function $f(\mathbf{x})$ be convex and satisfy the (first-order) β-Coordinate Lipschitz condition, and admit a minimizer $\mathbf{x}^*$. Then*

$$E_{\xi^k}[f(\mathbf{x}^{k+1})] - f(\mathbf{x}^*) \leq \frac{n}{n+k+1}\left(\frac{1}{2}\|\mathbf{x}^0 - \mathbf{x}^*\|_\beta^2 + f(\mathbf{x}^0) - f(\mathbf{x}^*)\right),$$

where random vector $\xi_{k-1} = (i^0, i^1, ..., i^{k-1})$ and norm-square $\|\mathbf{x}\|_\beta^2 = \sum_j \beta_j x_j^2$.

(RABCDlls,m, RABCDglls.m, RandBCDlls.m, RandBCDglls.m of Chapter 8)

Stochastic-Gradient-Method for Minimizing a Large-Sum of Functions

In many applications, the objective value is partially determined by decision makers and partially determined by "Nature".

$$\begin{array}{lll} (OPT) & \min_{\mathbf{x}} & f(\mathbf{x},\omega) \\ & \text{s.t.} & \mathbf{c}(\mathbf{x},\omega) \in K \subset R^m. \end{array} \tag{9}$$

where ω represents uncertain data and $\mathbf{x} \in R^n$ is the decision vector, and K is a constraint set.

For deterministic optimization, we assume ξ is known and fixed. In reality, we may have

- the (exact) probability distribution ξ of data ω.
- the sample distribution and/or few moments of data ω.
- knowledge of ω belonging to a given uncertain set U.

In the following we consider the unconstrained case.

Stochastic Optimization and Stochastic Gradient Descent (SGD) Methods

$$\min_{\mathbf{x}} \quad F(\mathbf{x}) := \mathsf{E}_{\xi}[f(\mathbf{x}, \omega)].$$

Large-Sum of Functions – Sample Average Approximation (SAA):

$$\min_{\mathbf{x}} \quad F_M(\mathbf{x}) := \tfrac{1}{M} \textstyle\sum_{i=1}^{M} f(\mathbf{x}, \omega^i).$$

Two Approaches:

- Sample-First and Iterate-Second, in particular, SAA: collect enough examples then search a solution of an approximated deterministic optimization problem. The computation of the gradient vector:

$$\nabla F_M(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} \nabla f(\mathbf{x}, \omega^i) \quad \text{and} \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \nabla F_M(\mathbf{x}^k).$$

- Sample and Iterate Concurrently – SGD: collect a sample set S^k of few samples of ω at iteration k:

$$\hat{\mathbf{g}}^k = \frac{1}{|S^k|} \sum_{i \in S^k} \nabla f(\mathbf{x}^k, \omega^i) \quad \text{and} \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \hat{\mathbf{g}}^k.$$

(SGDlls.m of Chapter 8)

Key Questions: how many samples are sufficient for an ϵ approximate solution to the original stochastic optimization problem. This is the information/sample complexity issue in optimization.

Special Case: Online Gradient-Descent and Regret

Consider more general convex minimization

$$f^* := \quad \min_{\mathbf{x}} \quad F_M(\mathbf{x}) := \tfrac{1}{M}\textstyle\sum_{i=1}^M f^i(\mathbf{x}).$$

where each f^i is a convex function.

OGD: Starting $\mathbf{x}^1$, for $i = 1$ to M, do

$$\hat{\mathbf{g}}^i = \nabla f^i(\mathbf{x}^i) \quad \text{and} \quad \mathbf{x}^{i+1} = \mathbf{x}^i - \alpha^k \hat{\mathbf{g}}^i.$$

Provable Result: for a wide range problems,

$$\frac{1}{M}\sum_{i=1}^{M} f^i(\mathbf{x}^i) \le f^* + O(\frac{1}{\sqrt{M}}).$$

Under certain conditions, the regret/error can be further reduced!

SGD and its Advantages

Apply SGD with one ω^k sampled uniformly at iteration k:

$$\hat{\mathbf{g}}^k = \nabla f(\mathbf{x}^k, \omega^k) \quad \text{and} \quad \mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \hat{\mathbf{g}}^k;$$

with the step size rule:

$$\alpha^k \to 0 \quad \text{and} \quad \left(\sum_{k=0}^{\infty} \alpha^k \right) \to \infty \quad \text{(e.g., } \alpha_k = O(k^{-1})\text{).}$$

- A great technology to potentially reduce the computation complexity – need fewer samples at the beginning.
- Potentially only select important and sensitive samples – learn where to sample.
- Dynamically incorporate new empirical observations to tune-up the probability distribution.
- May help to escape from the Saddle Points!

Various **Variance Reduction Techniques** were developed in Stochastic SGD! For example, let $\mathbf{d}^k = (1-\beta)\hat{\mathbf{g}}^k + \beta \mathbf{d}^{k-1}$ for some parameter β and then $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k \mathbf{d}^k$.

Case I: Variance Reduction in Stochastic Value Iteration for MDP

Let $\mathbf{y} \in \mathbf{R}^m$ represent the cost-to-go values of the m states, ith entry for ith state, of a given policy. The MDP problem entails choosing the fixed-point value vector $\mathbf{y}^*$ such that it satisfies:

$$y_i^* = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*\}, \ \forall i.$$

The Value-Iteration (VI) Method is, starting from any $\mathbf{y}^0$,

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\}, \ \forall i.$$

If the initial $\mathbf{y}^0$ is strictly feasible for state i, that is, $y_i^0 < c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0, \ \forall j \in \mathcal{A}_i$, then y_i^k would be increasing in the VI iteration for all i and k.

The computation work for state i at iteration k, is to compute $\mathbf{p}_j^T \mathbf{y}^k = \mu_j(\mathbf{y}^k)$ for each $j \in \mathcal{A}_i$. This needs $O(m)$ operations.

Could we approximate $\mu_j(\mathbf{y}^k)$ by sampling?

Motivations

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T\mathbf{y}^k$ by stochastic sampling,
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T\mathbf{y}^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.
- Since randomization is introduced in the algorithm, the iterative solution sequence becomes a random sequence.
- One can analyze this performance using Hoeffding's inequality and classic results on contraction properties of value iteration. Moreover, we improve the final result using Variance Reduction and Monotone Iteration.
- Variance Reduction enables us to update the values so that the needed number of samples is decreased from iteration to iteration.

Variance Reduction in Stochastic Value Iteration for MDP

We carry out the VI iteration as:

$$y_i^{k+1} = \min_{j\in\mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k + \gamma \mathbf{p}_j^T(\mathbf{y}^k - \tilde{\mathbf{y}}^k)\},\ \forall i,$$

where $\tilde{\mathbf{y}}^k$ is updated at the geometric pace as before. Or compute once a while for a hash vector

$$\tilde{c}_j^k = c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k,\ \forall j$$

and do

$$y_i^{k+1} = \min_{j\in\mathcal{A}_i} \{\tilde{c}_j^k + \gamma \mathbf{p}_j^T(\mathbf{y}^k - \tilde{\mathbf{y}}^k)\},\ \forall i.$$

Then we only need to approximate

$$\mathbf{p}_j^T(\mathbf{y}^k - \tilde{\mathbf{y}}^k) = \mu_j(\mathbf{y}^k - \tilde{\mathbf{y}}^k).$$

Since $\mathbf{y}^* \geq \mathbf{y}^k \geq \tilde{\mathbf{y}}^k$ during the period of k to $2k$ and $(\mathbf{y}^k - \tilde{\mathbf{y}}^k)$ monotonically converges to zero, the norm of $(\mathbf{y}^k - \tilde{\mathbf{y}}^k)$ becomes smaller and smaller so that only a constant number of samples are needed to estimate the mean for desired accuracy, which leads to a geometrically convergent algorithm with high probability.

Near-Optimal Randomized Value-Iteration Result

Few computation and sample complexity results based on Variance Reduction:

- Knowing $\mathbf{p}_j$:

$$O\left((mn + \frac{n}{(1-\gamma)^3}) \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

- Computation and sample complexity on the pure generative model:

$$O\left(\frac{n}{(1-\gamma)^3\epsilon^2} \log(\frac{1}{\delta})\right)$$

  to compute an ϵ-optimal policy with probability at least $1-\delta$.

- Sample complexity lower bound: $O\left(\frac{n}{(1-\gamma)^3\epsilon^2}\right)$.
- The method is also extended to computing ϵ-optimal policies for finite-horizon MDP with a generative model and provide a nearly matching sample complexity lower bound.

S[ICML 2017] and [NIPS 2018].

Case II: Online Linear Programming (OLP) Revisited

Primal

$$\begin{array}{ll} \max & \mathbf{r}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} \le \mathbf{b} \\ & \mathbf{0} \le \mathbf{x} \le \mathbf{e} \end{array}$$

Dual

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \mathbf{e}^\top \mathbf{s} \\ \text{s.t.} & A^\top \mathbf{p} + \mathbf{s} \ge \mathbf{r} \\ & \mathbf{p} \ge \mathbf{0}, \mathbf{s} \ge \mathbf{0} \end{array}$$

where the decision variables are $\mathbf{x} \in \mathcal{R}^n$, $\mathbf{p} \in \mathcal{R}^m, \mathbf{s} \in \mathcal{R}^n$

Denote the offline primal/dual optimal solution as $\mathbf{x}^* \in \mathcal{R}^n, \mathbf{p}^*_n \in \mathcal{R}^m, \mathbf{s}^* \in \mathcal{R}^n$

LP duality/complementarity tells that for $j = 1, ..., n,$

$$x_j^* = \begin{cases} 1, & r_j > \mathbf{a}_j^\top \mathbf{p}_n^* \\ 0, & r_j < \mathbf{a}_j^\top \mathbf{p}_n^* \end{cases}$$

x_j^* may take a fractional value when $r_j = \mathbf{a}_j^\top \mathbf{p}_n^*$.

Equivalent Form of the Dual Problem (I)

The dual objective is a large-sum of functions:

$$\begin{array}{ll} \text{min} & \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^n s_j \\ \text{s.t.} & s_j \geq r_j - \mathbf{a}_j^\top \mathbf{p}, \;\; j = 1, ..., n; \;\; \mathbf{p}, \mathbf{s} \geq \mathbf{0} \end{array}$$

Equivalently, by removing s_j's, we have

$$\min_{\mathbf{p} \geq \mathbf{0}} \; \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

where $(\cdot)^+$ is the ReLu function. Normalizing the objective, the large-sum functions become SAA:

$$\min_{\mathbf{p} \geq \mathbf{0}} \; f_n(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{n} \sum_{j=1}^{n} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

Implication for online LP when orders coming randomly:

- At time t, one can resolve $f_t(\mathbf{p})$ (based on all the observed samples) to obtain $\mathbf{p}_t^*$ and decide x_t

$$\min_{\mathbf{p} \geq \mathbf{0}} \; f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t} \sum_{j=1}^{t} \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

- Simply apply one step of Stochastic Sub-Gradient Projection Method to decide x_t and update $\mathbf{p}$.

The Simple and Fast Iterative OLP Algorithm

Instead of finding the optimal $\mathbf{p}_t^*$, we perform stochastic sub-gradient descent based on the newly arrived order t in minimizing

$$\min_{\mathbf{p}\geq 0} \; f_t(\mathbf{p}) := \mathbf{d}^\top \mathbf{p} + \frac{1}{t}\sum_{j=1}^{t}\left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+$$

At time t, the sub-gradient constructed from the new observation is

$$\begin{aligned} \nabla_{\mathbf{p}}\left(\mathbf{d}^\top \mathbf{p} + \left(r_t - \mathbf{a}_t^\top \mathbf{p}\right)^+\right)\Bigg|_{\mathbf{p}=\mathbf{p}_t} &= \mathbf{d} - \mathbf{a}_t I(r_t > \mathbf{a}_t^\top \mathbf{p})\Big|_{\mathbf{p}=\mathbf{p}_t} \\ &= \mathbf{d} - \mathbf{a}_t x_t \end{aligned}$$

where $\mathbf{p}_t$ is the current dual price vector at time t.

Simple Online (SO) Algorithm for Solving (Binary) Online LP I

- Input: $\mathbf{d} = \mathbf{b}/n$ and initialize $\mathbf{p}_1 = \mathbf{0}$
- For $t = 1, 2, ..., n$ do

$$x_t = \begin{cases} 1, & \text{if } r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{if } r_t \le \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$$

- Then compute

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \alpha_t \left(\mathbf{a}_t x_t - \mathbf{d} \right) \\ \mathbf{p}_{t+1} &:= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$

- Return $\mathbf{x} = (x_1, ..., x_n)$

This is Sample without Replacement Implementation of Stochastic Gradient Method with one Cycle only, where the primal decision is made "on the fly".

(fastOLP.m and fastOLPadap.m of Chapter 8)

Simple Online (SO) Algorithm for Solving (Binary) Online LP II

- The algorithm is a first-order online algorithm and it does not involve any matrix inversion.
- It does not need even to store the data, the total number of operations is the number of nonzero entries of all input data.
- α_t is the step size and it is chosen to be $\frac{1}{\sqrt{n}}$ (or $\frac{1}{\sqrt{t}}$) in the following analyses
- The algorithm does not require any prior knowledge besides $\mathbf{d}$, the average inventory vector.
- May add “adaptiveness (action-history-dependent)” and/or “boosting (sample replacement)” ideas to improve effectiveness
- May apply the Mirror-Descent and other first-order methods

The algorithm works for both the stochastic input model and the random permutation model following where the performance is guaranteed in expectation.

Non-Stationary Data Input?

Summary of the First-Order Methods

- Good global convergence property (e.g. starting from any (feasible) solution under mild technical assumption...).
- Simple to implement and the computation cost is mainly compute the numerical gradient and matrix-vector product; suitable on GPU
- Maybe difficult to decide step-size or learning rate
- The convergence speed can be slow: not suitable for high accuracy computation, certain accelerations available; but Gradient-scaling or Preconditioning will help
- Can only guarantee converging to a first-order KKT solution, may need to add perturbation to escape local minimums.
- Dimension reduced stochastic gradient and randomized block-coordinate are commonly used for solving large-scale problems.