Optimization Algorithms Review and Project Discussions

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

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

Optimization Algorithms

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 scalar $\{x^k\}$ 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.

Generic Algorithms for Minimization and Global Convergence Theorem

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

Theorem 1 *Let A be an "algorithmic mapping" defined over set X, and let sequence $\{\mathbf{x}^k\}$, starting from a given point $\mathbf{x}^0$, be generated from*

$$\mathbf{x}^{k+1} \in A(\mathbf{x}^k).$$

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, $z(\mathbf{y}) \le z(\mathbf{x})$ for all $\mathbf{y} \in A(\mathbf{x})$;

iii) the mapping A is closed at points outside S.

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

A mapping is closed if $\mathbf{y}^k = A(\mathbf{x}^k) \rightarrow \bar{\mathbf{y}}$ and $\mathbf{x}^k \rightarrow \bar{\mathbf{x}}$ implies $\bar{\mathbf{y}} = A(\bar{\mathbf{x}})$ (mostly proved by contradiction).

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 length, 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.

(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

- 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 number γ such that

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

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

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

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

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

  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.

Golden-Section Method for one-dimensional search - linear convergence rate 0.618

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

Bi-Section Method for one-dimensional search - linear convergence rate 0.5

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

Newton's method: superior local quadratic (order-two) convergence, but may fail globally.

Most algorithm analyses are based on solving various Lipschitz conditions/constants.

All algorithms allow some inexactness in numerical computation.

First-Order Algorithms: the Steepest Descent Method

Let f be a differentiable function and assume we can compute (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 KKT solution 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 $\mathbf{x}^k$. The number $\alpha^k \geq 0$, called step-size, is chosen "appropriately," namely to satisfy

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

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

Now, if $\nabla f(\mathbf{x}^{k+1}) \neq \mathbf{0}$, then $-\nabla f(\mathbf{x}^{k+1})$ is a direction of descent at $\mathbf{x}^{k+1}$; in fact, it is the direction of steepest descent. The convergence speed of SDM is arithmetic with $\frac{1}{\epsilon^2}$ in general, and $\frac{1}{\epsilon}$ in the convex case, for desired accuracy ϵ.

Many different step-size rules and direction modifications were proposed.

SDM Variants I

- The Barzilai and Borwein rule: Let $\Delta_x^k = \mathbf{x}^k - \mathbf{x}^{k-1}$ and $\Delta_g^k = \nabla f(\mathbf{x}^k) - \nabla f(\mathbf{x}^{k-1})$,

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

- The fixed step-size rule: $\alpha^k = \frac{1}{\beta}$ for objective functions being the (first-order) β-Lipschitz; that is, for any two vectors $\mathbf{x}$ and $\mathbf{d}$,

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

- Doubling/Halving rule: stat at a good step-size guess α:

Case 1: If $\alpha \le \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$ then $\alpha \leftarrow 2\alpha$, stop as soon as the inequality is reversed and return the latest α with $\alpha \le \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$;

Case 2: Otherwise $\alpha \leftarrow \alpha/2$; stop as soon as $\alpha \le \frac{2(f(\mathbf{x}^k)-f(\mathbf{x}^k+\alpha\mathbf{d}^k))}{\|\mathbf{d}^k\|^2}$ and return it.

SDM Variants II

- The Accelerated Steepest Descent Method (ASDM): a carefully designed modification of two consecutive SDM step solutions for for objective functions being the (first-order) β-Lipschitz, where the convergence speed of ASDM is arithmetic with $\frac{1}{\epsilon^{1/2}}$.
- The Conjugate Gradient (CG) Method: an intermediate SDM and Newton's method obtained by successively selecting conjugate-gradient directions as the method progresses.
- The Parallel Tangent Method (PARTAN): a complete cycle consists of taking two steepest descent steps and then searching along the line connecting the starting point and the point obtained after the two SDM steps.
- Stochastic (sub)Gradient Method, application to online linear programming.
- The Quasi-Newton (QN) Method: an essentially Newton's method with a on-line learning process of Hessian via rank-one or rank-two update.

All of the latter three methods converge in finite number of steps for solving convex quadratic optimization.

Second-Order (SO) Algorithms: Newton's Method with a Step-Size

For unconstrained optimization, the iteration is given by

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

where z in the pure Newton's method $\alpha^k = 1$.

Convergence: quadratically if the starting point is close enough AND the Hessian matrix is non-singular at every iterate and the limit point. It converges in one step if the function is quadratic and the Hessian matrix is non-singular.

A step-size would work for strictly convex optimization, since $(\nabla^2 f(\mathbf{x}^k))^{-1}$ is positive definite so that $-(\nabla^2 f(\mathbf{x}^k))^{-1} \nabla f(\mathbf{x}^k)$ is a descent direction.

More generally, a modified Newton method would be

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha^k (\nabla^2 f(\mathbf{x}^k) + \gamma^k I)^{-1} \nabla f(\mathbf{x}^k),$$

for some positive number γ^k such that $\nabla^2 f(\mathbf{x}^k) + \gamma^k I$ is positive definite for descent property.

The convergence analysis of SO algorithms is typically associated with the so-called second-order

β-Lipschitz condition: for any two points $\mathbf{x}$ and vector $\mathbf{d}$

$$\|\nabla f(\mathbf{x}+\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\mathbf{d}\| \leq \beta\|\mathbf{d}\|^2,$$

which implies

$$f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x}) \leq \nabla f(\mathbf{x})^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} + \frac{\beta}{3}\|\mathbf{d}\|^3.$$

Note that a function may not be Lipschitz everywhere, but Lipschitz when $\mathbf{x}$ is bounded.

Thus, one can solve a sequence of cubic minimization:

$$\min_{\alpha,\mathbf{d}} \ \nabla f(\mathbf{x}^k)^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x}^k)\mathbf{d}, \text{ s.t. } \|\mathbf{d}\| \leq \alpha$$

which is called the spherical “Trust-Region” method, which would be discussed next.

The convergence speed is arithmetic with $\frac{1}{\epsilon^{1.5}}$ in general, and $\log(\frac{1}{\epsilon})$ in the convex case, for desired accuracy ϵ.

Spherical Trust-Region Method for Minimizing Lipschitz $f(\mathbf{x})$

Recall the second-order β-Lipschitz condition: for any two points $\mathbf{x}$ and $\mathbf{d}$

$$\|\mathbf{g}(\mathbf{x}+\mathbf{d}) - \mathbf{g}(\mathbf{x}) - \nabla\mathbf{g}(\mathbf{x})\mathbf{d}\| \le \beta\|\mathbf{d}\|^2,$$

where $\mathbf{g}(\mathbf{x}) = \nabla f(\mathbf{x})$ and $\nabla\mathbf{g}(\mathbf{x}) = \nabla^2 f(\mathbf{x})$. It implies

$$f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x}) \le \nabla f(\mathbf{x})^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} + \frac{\beta}{3}\|\mathbf{d}\|^3.$$

$$\begin{array}{ll}
 & f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x}) - \nabla f(\mathbf{x})^T\mathbf{d} - \frac{1}{2}\mathbf{d}^T\nabla^2 f(\mathbf{x})\mathbf{d} \\
= & \int_0^1 \mathbf{d}^T(\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}))\mathrm{dt} - \frac{1}{2}\mathbf{d}^{\mathrm{T}}\nabla^2\mathrm{f}(\mathbf{x})\mathbf{d} \\
= & \int_0^1 \mathbf{d}^T\left(\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})(t\mathbf{d})\right)\mathrm{dt} \\
\le & \int_0^1 \|\mathbf{d}\|\|\nabla f(\mathbf{x}+t\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})(t\mathbf{d})\|\mathrm{dt} \\
\le & \int_0^1 \|\mathbf{d}\|\beta\|t\mathbf{d}\|^2\mathrm{dt} \text{ (by 2nd-order -Lipschitz condition)} \\
= & \beta\|\mathbf{d}\|^3\int_0^1 t^2\mathrm{dt} = \frac{\beta}{3}\|\mathrm{d}\|^3.
\end{array}$$

The second-order method, at the kth iterate, would let $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k$ where

$$\begin{array}{rll} \mathbf{d}^k = & \arg\min_{\mathbf{d}} & (\mathbf{c}^k)^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T Q^k \mathbf{d} + \frac{\beta}{3}\alpha^3 \\ & \text{s.t.} & \|\mathbf{d}\| \le \alpha, \end{array}$$

with $\mathbf{c}^k = \nabla f(\mathbf{x}^k)$ and $Q^k = \nabla^2 f(\mathbf{x}^k)$. One typically fixed α to a "trusted" radius α^k so that it becomes a sphere-constrained problem (the inequality is normally active if the Hessian is non PSD):

$$(Q^k + \lambda^k I)\mathbf{d}^k = -\mathbf{c}^k,\ (Q^k + \lambda^k I) \succeq \mathbf{0},\ \|\mathbf{d}^k\|_2^2 = (\alpha^k)^2.$$

For fixed α^k, the method is generally called trust-region method.

The Trust-Region can be ellipsoidal such as $\|S\mathbf{d}\| \le \alpha$ where S is a PD diagonal scaling matrix.

Convergence Speed of the Spherical Trust-Region Method

Is there a trusted radius such that the method converging? A simple choice would fix $\alpha^k = \sqrt{\epsilon}/\beta$. Then from reduction (**??**)

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \le -\frac{\lambda^k}{2}\|\mathbf{d}^k\|^2 + \frac{\beta}{3}(\alpha^k)^3 = -\frac{\lambda^k(\alpha^k)^2}{2} + \frac{\beta}{3}(\alpha^k)^3 = -\frac{\lambda^k \epsilon}{2\beta^2} + \frac{\epsilon^{3/2}}{3\beta^2}.$$

Also

$$\begin{aligned}\|\mathbf{g}(\mathbf{x}^{k+1})\| &= \|\mathbf{g}(\mathbf{x}^{k+1}) - (\mathbf{c}^k + Q^k\mathbf{d}^k) + (\mathbf{c}^k + Q^k\mathbf{d}^k)\| \\ &\le \|\mathbf{g}(\mathbf{x}^{k+1}) - (\mathbf{c}^k + Q^k\mathbf{d}^k)\| + \|(\mathbf{c}^k + Q^k\mathbf{d}^k)\| \\ &\le \beta\|\mathbf{d}^k\|^2 + \lambda^k\|\mathbf{d}^k\| = \beta(\alpha^k)^2 + \lambda^k\alpha^k = \tfrac{\epsilon}{\beta} + \tfrac{\lambda^k\sqrt{\epsilon}}{\beta}.\end{aligned}$$

Thus, one can stop the algorithm as soon as $\lambda^k \le \sqrt{\epsilon}$ so that the inequality becomes $\|\mathbf{g}(\mathbf{x}^{k+1})\| \le \frac{2\epsilon}{\beta}$ and the function value is decreased at least $-\frac{\epsilon^{1.5}}{6\beta^2}$. Furthermore, $|\lambda_{min}(\nabla \mathbf{g}(\mathbf{x}^k))| \le \lambda^k = \sqrt{\epsilon}$.

Theorem 2 *Let the objective function* $p^* = \inf\ f(\mathbf{x})$ *be finite. Then in* $\frac{O(\beta^2(f(\mathbf{x}^0)-p^*))}{\epsilon^{1.5}}$ *iterations of the trust-region method, the norm of the gradient vector is less than* ϵ *and the Hessian is* $\sqrt{\epsilon}$*-positive semidefinite, where each iteration solves a spherical-constrained quadratic minimization discussed earlier.*

Adaptive Spherical Trust-Region Method

One can treat α as a variable in

$$\begin{array}{rll} \mathbf{d}^k = & \arg\min_{(\mathbf{d},\alpha)} & (\mathbf{c}^k)^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T Q^k \mathbf{d} + \frac{\beta}{3}\alpha^3 \\ & \text{s.t.} & \|\mathbf{d}\| \le \alpha. \end{array}$$

Then, the optimality conditions of this sub-problem would be

$$(Q^k + \lambda I)\mathbf{d}^k = -\mathbf{c}^k,\ (Q^k + \lambda I) \succeq \mathbf{0},\ \|\mathbf{d}\|_2^2 = \alpha^2,$$

and $\alpha = \frac{\lambda}{\beta}$. Thus, let $\mathbf{d}(\lambda) = -(Q^k + \lambda I)^{-1}\mathbf{c}^k$, the problem becomes finding the root λ of

$$\|\mathbf{d}(\lambda)\|^2 - \frac{\lambda^2}{\beta^2} = 0,$$

where $\lambda \ge -\lambda_{min}(Q^k) > 0$ (assume that the current Hessian is not PSD yet), as in the Hybrid of Bisection and Newton method discussed earlier in $\log\log(1/\epsilon)$ arithmetic operations.

In practice, even β is unknown, one can forward/backward choose λ such as the objective function is reduced by a sufficient quantity, and there is no need to find the exact root.

Relation to Quadratic Regularization/Proximal-Point Method

One can also interpret the Spherical Trust-Region method as the Quadratic Regularization Method

$$\mathbf{d}^k(\lambda) = \quad \arg\min_{\mathbf{d}} \quad (\mathbf{c}^k)^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T Q^k \mathbf{d} + \frac{\lambda}{2}\|\mathbf{d}\|^2$$

where parameter λ makes $(Q^k + \lambda I) \succeq \mathbf{0}$. Then consider the one-variable function

$$\phi(\lambda) := f(\mathbf{x}^k + \mathbf{d}^k(\lambda))$$

and do one-variable minimization of $\phi(\lambda)$ over λ. Then let λ^k be a minimizer and $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k(\lambda^k)$.

Thus, based on the earlier analysis, we must have at least

$$f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) \le -\frac{\epsilon^{1.5}}{6\beta^2}$$

for some (local) Lipschitz constant β of the objective function.

Note that the algorithm needs to estimate only the minimum eigenvalue, $\lambda_{min}(Q^k)$, of the Hessian. One heuristic is to let λ^k decreases geometrically and do few possible line-search steps.

Dimension-Reduced Second-Order Method with Trust Region: one dimension

Let $H^k = \nabla^2 f(\mathbf{x}^k)$, $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$, $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \mathbf{g}^k$ or $\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha \mathbf{g}^k/\|\mathbf{g}^k\|$ for some α. Let $\bar{\mathbf{g}}^k = \mathbf{g}^k/\|\mathbf{g}^k\|$. Then, for the second expression, one can solve the one-dimensional trust-region problem: $\alpha^k = \arg\min_{\alpha \in R} \; -\|\mathbf{g}^k\|\alpha + \frac{q^k}{2}\alpha^2$, subject to spherical trust-region (or interval) $\alpha^2 \le (\epsilon/\beta)^2$, where $q^k = (\bar{\mathbf{g}}^k)^T H^k \bar{\mathbf{g}}^k$. The optimality conditions of this sub-problem are $\lambda^k \ge 0$ and

$$(q^k + \lambda^k)\alpha^k = \|\mathbf{g}^k\|, \; (q^k + \lambda^k) \ge 0, \; \lambda^k(\sqrt{\epsilon}/\beta - \alpha^k) = 0.$$

As long as $\lambda^k \ge \sqrt{\epsilon}$, the objective function would be reduced by the same quantity as that in the full-dimensional case. Otherwise, if further $q^k \le \sqrt{\epsilon}$, we must have $\|\mathbf{g}^k\| \le \frac{2\epsilon}{\beta}$.

Now what happens when $\lambda^k < \sqrt{\epsilon}$ and $q^k > \sqrt{\epsilon}$ in the sub-problem? Denote by $\mathbf{d}^k = -\alpha^k \bar{\mathbf{g}}^k$, $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{d}^k$ and $\mathbf{g}^{k+1} = \nabla f(\mathbf{x}^{k+1})$. Then

$$\mathbf{g}^{k+1} = (\mathbf{g}^{k+1} - \mathbf{g}^k - H^k \mathbf{d}^k) + (\mathbf{g}^k + H^k \mathbf{d}^k)$$

and

$$
\begin{aligned}
(\bar{\mathbf{g}}^k)^T\mathbf{g}^{k+1} &= (\bar{\mathbf{g}}^k)^T(\mathbf{g}^{k+1}-\mathbf{g}^k-H^k\mathbf{d}^k)+(\bar{\mathbf{g}}^k)^T(\mathbf{g}^k+H^k\mathbf{d}^k)\\
&= (\bar{\mathbf{g}}^k)^T(\mathbf{g}^{k+1}-\mathbf{g}^k-H^k\mathbf{d}^k)+\|\mathbf{g}^k\|-q^k\alpha^k\\
&\le (\bar{\mathbf{g}}^k)^T(\mathbf{g}^{k+1}-\mathbf{g}^k-H^k\mathbf{d}^k)+\lambda^k\alpha^k.
\end{aligned}
$$

which implies that (noting $\|\bar{\mathbf{g}}^k\|=1$):
$|(\bar{\mathbf{g}}^k)^T\mathbf{g}^{k+1}| \le \|\bar{\mathbf{g}}^k\|\cdot\|(\mathbf{g}^{k+1}-\mathbf{g}^k-H^k\mathbf{d}^k)\|+\frac{\epsilon}{\beta}\le\beta(\alpha^k)^2+\frac{\epsilon}{\beta}=\frac{2\epsilon}{\beta}.$

Proposition 1 *Let the objective function $p^*=\inf\ f(\mathbf{x})$ be finite. Then in $\frac{O(\beta^2(f(\mathbf{x}^0)-p^*))}{\epsilon^{1.5}}$ iterations of the one-dimensional trust-region method, the absolute-value of a gradient vector norm or the projection of the new gradient vector onto the current gradient subspace is less than $O(\epsilon)$. Furthermore, the Hessian is $\sqrt{\epsilon}$-positive semidefinite on the current gradient subspace.*

Note that if either $\|\mathbf{g}^{k+1}\|$ or $\|\mathbf{g}^k\|$ is less than ϵ, then the algorithm stops. Otherwise, $\mathbf{g}^{k+1}$ must be almost orthogonal to $\mathbf{g}^k$ which has created a new search direction very different from the previous one.

One can also try $\mathbf{x}^{k+1}=\mathbf{x}^k-\alpha^g\bar{\mathbf{g}}^k+\alpha^u\mathbf{u}^k$ where $\mathbf{u}^k$ is a random vector $\mathbf{u}^k\in N(\mathbf{0},I-\bar{\mathbf{g}}^k(\bar{\mathbf{g}}^k)^T)$ such that $E[(\bar{\mathbf{g}}^k)^T\mathbf{u}^k]=0$ and $E[\mathbf{u}^k(\mathbf{u}^k)^T+\bar{\mathbf{g}}^k(\bar{\mathbf{g}}^k)^T]=I$ and (α^g,α^u) is decided by solving a two-dimensional trust-region subproblem. Or choose $\mathbf{u}^k$ as the momentum direction $(\mathbf{x}^k-\mathbf{x}^{k-1})$ as follows.

Dimension-Reduced Second-Order Method with Trust Region: two-dimension

Let $H^k = \nabla^2 f(\mathbf{x}^k)$, $\mathbf{d}^k = \mathbf{x}^k - \mathbf{x}^{k-1}$ and $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$, and

$$Q^k = \begin{pmatrix} (\mathbf{g}^k)^T H^k \mathbf{g}^k & -(\mathbf{d}^k)^T H^k \mathbf{g}^k \\ -(\mathbf{d}^k)^T H^k \mathbf{g}^k & (\mathbf{d}^k)^T H^k \mathbf{d}^k \end{pmatrix} \in S^2, \mathbf{c}^k = \begin{pmatrix} -\|\mathbf{g}^k\|^2 \\ (\mathbf{g}^k)^T \mathbf{d}^k \end{pmatrix} \in R^2.$$

Then, similar to the full-dimensional Spherical Trust-Region, one can construct a 2-dimensional trust-region quadratic model:

$$\alpha^k(\lambda^k) = \quad \arg\min_{\alpha \in R^2} \quad (\mathbf{c}^k)^T \alpha + \tfrac{1}{2} \alpha^T Q^k \alpha + \tfrac{\lambda^k}{2} \|\alpha\|^2$$

where parameter λ^k makes $(Q^k + \lambda^k I) \succeq \mathbf{0}$. Finally let

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_1^k \mathbf{g}^k + \alpha_2^k \mathbf{d}^k.$$

Again, 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 more accurate difference approximation between two gradients.

A Path-Following Algorithm for Unconstrained Optimization I

For any $\mu > 0$ consider the (unique) optimal solution $\mathbf{x}(\mu)$ for problem

$$\mathbf{x}(\mu) = \arg\min\ f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x}\|^2,$$

and they form a path down to $\mathbf{x}(0)$ and satisfy gradient equations with parameter μ:

$$\mathbf{g}(\mathbf{x}) + \mu\mathbf{x} = \mathbf{0}, \quad \text{with} \quad \mu = \mu^k > 0. \tag{1}$$

Let the approximation path error at $\mathbf{x}^k$ with $\mu = \mu^k$ be

$$\|\mathbf{g}(\mathbf{x}^k) + \mu^k\mathbf{x}^k\| \le \frac{1}{2\beta}\mu^k.$$

Then, we like to compute a new iterate $\mathbf{x}^{k+1}$, using Newton's method with $\mathbf{x}^k$ as an initial solution, such that

$$\|\mathbf{g}(\mathbf{x}^{k+1}) + \mu^{k+1}\mathbf{x}^{k+1}\| \le \frac{1}{2\beta}\mu^{k+1}, \quad \text{where } 0 \le \mu^{k+1} < \mu^k.$$

If μ^k can be decreased at a geometric rate, independent of ϵ, and each update uses one Newton step, then this would lead to a linearly convergent algorithm.

Feasible FO Algorithms for Non-Negative Conic Optimization

$$\min\ f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{x} \geq \mathbf{0}.$$

- Interior-Point SDM on Logarithmic Barrier $\mu \sum_j \log(x_j)$ or Potential Function

$$2n \log(f(\mathbf{x}) - z) - \sum_j \log(x_j)$$

  where z is an objective lower bound.

- Interior-Point Affine-Scaling SDM: $\mathbf{d}^k = -(X^k)^2 \nabla f(\mathbf{x}^k)$, where $X^k = \text{diag}(\mathbf{x}^k)$; and the step-size is a certain fraction to the boundary and the Lipschitz constant, which ever is smaller. Its convergence speed is arithmetic with $\frac{1}{\epsilon^2}$ in general, and $\frac{1}{\epsilon}$ in the convex case, for desired accuracy ϵ.

- SDM Followed by Feasible-Region-Projection: Question in HW3
  - $\hat{\mathbf{x}}^{k+1} = \mathbf{x}^k - \frac{1}{\beta} \nabla f(\mathbf{x}^k)$
  - $\mathbf{x}^{k+1} = \text{Proj}_K(\hat{\mathbf{x}}^{k+1})$

Feasible FO Algorithms for SDP Conic Optimization

$$\min\ f(X) \quad \text{s.t.} \quad X \succeq \mathbf{0}.$$

- Interior-Point SDM with Logarithmic Barrier $\mu \log\det(X)$ or Potential Function

$$2n\log(f(X)-z) - \log\det(X)$$

  where z is an objective lower bound.

- Interior-Point Affine-Scaling SDM: $D^k = -X^k \nabla f(X^k) X^k$; and the step-size is a certain fraction to the boundary and the Lipschitz constant, which ever is smaller. Its convergence speed is arithmetic with $\frac{1}{\epsilon^2}$ in general, and $\frac{1}{\epsilon}$ in the convex case, for desired accuracy ϵ.

- SDM Followed by Feasible-Region-Projection (Question in HW3):
  - $\hat{X}^{k+1} = X^k - \frac{1}{\beta}\nabla f(X^k)$
  - $X^{k+1} = \text{Proj}_K(\hat{X}^{k+1})$

Feasible FO Algorithms for Optimization with Equality Constraints

$$\min \ f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b} \in R^m, \ \mathbf{x} \geq \mathbf{0} \in R^n.$$

- Reduced Gradient Method: at each step, consider the problem only in terms of the $n-m$ independent variables due to the linear equality constraints (which variables are called nonbasic variables in Linear Programming). Then the objective can be viewed as the independent variables, and its gradient with respect to independent variables is the reduced gradient/cost vector. Therefore, the problem simply becomes a non-negative conic problem.

- Sequential LP Method: Find a feasible and descent direction from the current feasible iterate $\mathbf{x}^k$:

$$\min_{\mathbf{d}} \ \nabla f(\mathbf{x}^k)\mathbf{d} \quad \text{s.t.} \quad A\mathbf{d} = \mathbf{0} \in R^m, \ \mathbf{x}^k + \mathbf{d} \geq \mathbf{0} \in R^n,$$

  then take a step-size along the minimal direction.

- Interior-Point SDM: minimizing the potential function with only equalty constraints and apply the Gradient-Projection SDM; and its convergence speed is arithmetic with $\frac{1}{\epsilon^2}$ in general, and $\frac{1}{\epsilon}$ in the convex case, for desired accuracy ϵ.

- Backtrack step-size rule similar to those discussed earlier.

SO Algorithms for Linearly Constrained Optimization: Interior-Point Methods

$$\min\ f(\mathbf{x}) \quad \text{s.t.} \quad A\mathbf{x} = \mathbf{b} \in R^m,\ \mathbf{x} \ge \mathbf{0} \in R^n.$$

Furthermore, let $f(\mathbf{x})$ be convex and satisfy the second-order β_α-Scaled Concordant Lipschitz condition: for any point $\mathbf{x} > \mathbf{0}$ (where the function does not need to be differentiable at the boundary)

$$\|X\left(\nabla f(\mathbf{x}+\mathbf{d}) - \nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x})\mathbf{d}\right)\| \le \beta_\alpha \mathbf{d}^T \nabla^2 f(\mathbf{x})\mathbf{d},\ \text{whenever}\ \|X^{-1}\mathbf{d}\| \le \alpha(<1).$$

This condition is specially suitable for analyzing interior-point methods described below.

Given a strictly feasible $(\mathbf{x} > \mathbf{0}, \mathbf{y}, \mathbf{s} = \nabla f(\mathbf{x}) - A^T\mathbf{y} > \mathbf{0})$, compute direction vectors $(\mathbf{d}_x, \mathbf{d}_y, \mathbf{d}_s)$:

$$\begin{array}{rcl} S\mathbf{d}_x + X\mathbf{d}_s & = & \mathbf{r} := \frac{\mathbf{x}^T\mathbf{s}}{n+\rho}\mathbf{e} - X\mathbf{s}, \\ A\mathbf{d}_x & = & \mathbf{0}, \\ \nabla^2 f(\mathbf{x})\mathbf{d}_x - A^T\mathbf{d}_y - \mathbf{d}_s & = & \mathbf{0}. \end{array}$$

Let

$$\theta = \frac{\alpha\sqrt{\min(XS\mathbf{e})}}{\|(XS)^{-1/2}\mathbf{r}\|}$$

where $\alpha \in (0\ 1)$ is a constant depending on β_α, and

$$\mathbf{x}^+ = \mathbf{x} + \theta \mathbf{d}_x, \quad \mathbf{y}^+ = \mathbf{y} + \theta \mathbf{d}_y, \quad \text{and} \quad \mathbf{s}^+ = \nabla f(\mathbf{x}^+) - A^T \mathbf{y}^+.$$

Then, $(\mathbf{x}^+ > \mathbf{0}, \mathbf{y}, \mathbf{s}^+ > \mathbf{0})$ is strictly feasible and the algorithm convergence speed is $\log \frac{1}{\epsilon}$. For linear programming, the speed is quadratically convergent once the iterate is close to the optimal solution set.

No need to understand complexity proofs of interior-point methods, but high level ideas and concepts such as central-path and potential functions!

The Lagrangian Function and Method

Consider

$$f^* := \min \quad f(\mathbf{x}) \quad \text{s.t.} \quad \mathbf{h}(\mathbf{x}) = \mathbf{0},\ \mathbf{x} \in X.$$

The Lagrangian function:

$$L(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{h}(\mathbf{x}),$$

and the Augmented Lagrangian function:

$$L_a(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T \mathbf{h}(\mathbf{x}) + \frac{\beta}{2}\|\mathbf{h}(\mathbf{x})\|^2.$$

Let the dual function be:

$$\phi(\mathbf{y}) = \min_{\mathbf{x} \in X} L_a(\mathbf{x}, \mathbf{y}); \tag{2}$$

and the dual problem

$$(f^* \geq)\phi^* := \max \quad \phi(\mathbf{y}). \tag{3}$$

In many cases, one can find $\mathbf{y}^*$ of dual problem (3), a unconstrained optimization problem; then go ahead to find $\mathbf{x}^*$ using (2).

- The Augmented Lagrangian Method: ($\frac{1}{\epsilon}$)

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}\in X} L_a(\mathbf{x}, \mathbf{y}^k), \text{ and } \mathbf{y}^{k+1} = \mathbf{y}^k - \beta\mathbf{h}(\mathbf{x}^{k+1}).$$

- The ADMM: ($\frac{1}{\epsilon}$)

$$\begin{aligned} \mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} L_a(\mathbf{x}_1, \mathbf{x}_2^k, \mathbf{y}^k), \\ \mathbf{x}_2^{k+1} &= \arg\min_{\mathbf{x}_2} L_a(\mathbf{x}_1^{k+1}, \mathbf{x}_2, \mathbf{y}^k), \\ \mathbf{y}^{k+1} &= \mathbf{y}^k - \beta\mathbf{h}(\mathbf{x}_1^{k+1}, \mathbf{x}_2^{k+1}). \end{aligned}$$

- The Multi-Block ADMM and Randomly Permuted Version

$$\begin{aligned} \mathbf{x}_1^{k+1} &= \arg\min_{\mathbf{x}_1} L_a(\mathbf{x}_1, ..., \mathbf{x}_n^k, \mathbf{y}^k), \\ ... \quad & ... \\ \mathbf{x}_n^{k+1} &= \arg\min_{\mathbf{x}_2} L_a(\mathbf{x}_1^{k+1}, ...\mathbf{x}_n, \mathbf{y}^k), \\ \mathbf{y}^{k+1} &= \mathbf{y}^k - \beta\mathbf{h}(\mathbf{x}_1^{k+1}, ..., \mathbf{x}_2^{k+1}). \end{aligned}$$

Block Coordinate Descent Method for Unconstrained Optimization

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

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

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

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) $\frac{1}{\epsilon}$.
- Aitken Double Sweep Method $\frac{1}{\epsilon}$.
- Gauss-Southwell Method $\frac{1}{\epsilon}$.
- Randomly-Permuted Cyclic Block Coordinate Descent (RCBCD) Method $\frac{1}{\epsilon}$.
- Randomized Block Coordinate Descent (RBCD) Method $\frac{1}{\epsilon}$.

Often, one does not minimize each coordinate exactly, but takes a gradient step.

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

Large-Sum of Functions, e.g., 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:

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

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.

SGD with Batch-Size One

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

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

- Works in general with the step size rule:

$$\alpha^k \to 0 \quad \text{and} \quad \left(\sum_{k=0}^{\infty} \alpha^k\right) \to \infty \quad \text{(e.g., } \alpha_k = O(k^{-1/2})\text{)}.$$

- Specifically, $\alpha_k = \frac{1}{\beta\sqrt{k}}$, where β is the largest $\|\hat{\mathbf{g}}^k\|$; and return $\bar{\mathbf{x}} = \frac{1}{K}\sum_{k=0}^{K-1} \mathbf{x}^k$ (L&Y pp292).
- 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.

Project-I: SNL

$$
\begin{aligned}
\|\mathbf{x}_i - \mathbf{x}_j\|^2 &= d_{ij}^2, \ \forall\,(i,j) \in N_x, \ i < j, \\
\|\mathbf{a}_k - \mathbf{x}_j\|^2 &= \hat{d}_{kj}^2, \ \forall\,(k,j) \in N_a,
\end{aligned} \tag{4}
$$

SDP relaxation for solving solve (4): Find a symmetric matrix $Z \in S^{d+n}$ such that

$$
\begin{aligned}
\min \quad & \mathbf{0} \bullet Z \\
\text{s.t.} \quad & Z_{1:d,1:d} = I, \\
& (\mathbf{0}; \mathbf{e}_i - \mathbf{e}_j)(\mathbf{0}; \mathbf{e}_i - \mathbf{e}_j)^T \bullet Z = d_{ij}^2, \ \forall\, i,j \in N_x, \ i < j, \\
& (\mathbf{a}_k; -\mathbf{e}_j)(\mathbf{a}_k; -\mathbf{e}_j)^T \bullet Z = \hat{d}_{kj}^2, \ \forall\, k,j \in N_a, \\
& Z \succeq \mathbf{0}.
\end{aligned} \tag{5}
$$

Also a simple nonlinear least squares (NLS) approach to solve (4):

$$
\min \quad \sum_{(ij)\in N_x} \left(\|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2\right)^2 + \sum_{(kj)\in N_a} \left(\|\mathbf{a}_k - \mathbf{x}_j\|^2 - d_{kj}^2\right)^2 \tag{6}
$$

P-I: Questions

- Run some randomly generated problems (in 1D, 2D and 3D) with few (2, 3 and 4) anchors and tens sensors to compare the SOCP, SDP, and NLS approaches.

- Create some noise data and solve the SDP relaxation to minimize
$\sum_{(ij)\in N_x} \left| \|\mathbf{x}_i - \mathbf{x}_j\|^2 - d_{ij}^2 \right| + \sum_{(kj)\in N_a} \left| \|\mathbf{a}_k - \mathbf{x}_j\|^2 - d_{kj}^2 \right|$

- Use the SDP solution $\bar{X} = [\bar{\mathbf{x}}_1,\ \bar{\mathbf{x}}_2,\ ....,\ \bar{\mathbf{x}}_n]$ of $\bar{Z}$ as the initial solution for model 6 and apply the Steepest Descent Method for a number steps. How is the final solution?

- Apply ADMM to the split nonlinear least squares:

$$\begin{array}{ll} \min & \sum_{(ij)\in N_x} \left[(\mathbf{x}_i - \mathbf{x}_j)^T(\mathbf{y}_i - \mathbf{y}_j) - d_{ij}^2\right]^2 + \sum_{(kj)\in N_a} \left[(\mathbf{a}_k - \mathbf{x}_j)^T(\mathbf{a}_k - \mathbf{y}_j) - d_{kj}^2\right]^2 \\ \text{s.t.} & \mathbf{x}_j - \mathbf{y}_j = \mathbf{0},\ \forall j. \end{array}$$

- Apply Steepest Descent and Feasible Projection Method to SDP Relaxation (Slide 9 of Lecture 12) by fixing one sensor at the origin, that is, only needs to localize $n-1$ sensors. Furthermore, if the location of other two sensors are known, then we can determine all other sensor locations.

Project-II: An Online Linear Programming/Resource Allocation Example

	order 1($t = 1$)	order 2($t = 2$)	.....	Inventory($\mathbf{b}$)
Price(π_t)	\$100	\$30	...	
Decision	x_1	x_2	...	
Pants	1	0	...	100
Shoes	1	0	...	50
T-shirts	0	1	...	500
Jacket	0	0	...	200
Socks	1	1	...	1000

P-II: Offline Formulation LP and CP

$$
\begin{array}{llll}
\text{LP} & \max & \pi^T\mathbf{x} & \\
& \text{s.t.} & A^T\mathbf{x} & \leq \mathbf{b}, \\
& & \mathbf{x} & \leq \mathbf{e}, \\
& & \mathbf{x} & \geq \mathbf{0},
\end{array}
\quad \text{or} \quad
\begin{array}{llll}
\text{CP} & \max & \pi^T\mathbf{x} + U(\mathbf{s}) & \\
& \text{s.t.} & A^T\mathbf{x} + \mathbf{s} & = \mathbf{b},\ \mathbf{s} \geq \mathbf{0} \\
& & \mathbf{x} & \leq \mathbf{e}, \\
& & \mathbf{x} & \geq \mathbf{0}.
\end{array}
$$

$U(\cdot)$, in the form $U(\mathbf{s}) = \sum_i u(s_i)$, is a strcitly concave (risk aversion) and increasing value function of the possible slack variables to value the uncertain revenue of remaining resources.

- Exponential $u(s_i) = w \cdot (1 - \exp(-as_i))$ for some positive constants a, w.
- Logarithmic $u(s_i) = w \cdot \log(s_i)$ or $u(s_i) = w \cdot \log(1 + s_i)$ for some positive constant w.
- Quadratic: $u(s_i) = \begin{cases} w \cdot (1 - (1 - s_i/w)^2) & 0 \leq s_i \leq w \\ w & s_i \geq w \end{cases}$ for some positive constant b.

P-II: Price Mechanism and SLPM and SCPM

The problem would be easy if there is an "ideal price" vector:

	Bid 1($t = 1$)	Bid 2($t = 2$)	.....	Inventory($\mathbf{b}$)	$\mathbf{p}^*$
Bid(π_t)	\$100	\$30	...		
Decision	x_1	x_2	...		
Pants	1	0	...	100	\$45
Shoes	1	0	...	50	\$45
T-shirts	0	1	...	500	\$10
Jackets	0	0	...	200	\$55
Hats	1	1	...	1000	\$15

Could such a "ideal price" vector be learned?

P-II: One-Time Learning Algorithm

We start with a simple

- Set $x_t = 0$ for all $1 \leq t \leq \epsilon n$;
- Solve the ϵ portion of the problem

$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\epsilon n} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\epsilon n} a_{it} x_t \leq \epsilon b_i & i = 1, ..., m \\ & 0 \leq x_t \leq 1 & t = 1, ..., \epsilon n \end{array}$$

  and get the optimal Lagrange/dual solution $\hat{\mathbf{p}}$;

- Determine the future allocation x_t as:

$$x_t = \begin{cases} 0 & \text{if } \pi_t \leq \hat{\mathbf{p}}^T \mathbf{a}_t \\ 1 & \text{if } \pi_t > \hat{\mathbf{p}}^T \mathbf{a}_t \end{cases}$$

  as long as $a_{it} x_t \leq b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ for all i; otherwise, set $x_t = 0$.

P-II: Dynamic/Online Learning Algorithm

In the dynamic price learning algorithm, we update the price at time ϵn, $2\epsilon n$, $4\epsilon n$, ..., till $2^k\epsilon \geq 1$.

At time $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price vector is the optimal Lagrange/dual solution to the following linear program:

$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq \frac{\ell}{n} b_i & i = 1, ..., m \ ; \\ & 0 \leq x_t \leq 1 & t = 1, ..., \ell \end{array}$$

and this price vector is used to determine the allocation for the next immediate period, which is doubled each update.

$$x_t = \begin{cases} 0 & \text{if } \pi_t \leq \hat{\mathbf{p}_\ell}^T \mathbf{a}_t \\ 1 & \text{if } \pi_t > \hat{\mathbf{p}_\ell}^T \mathbf{a}_t \end{cases}$$

as long as $a_{it} x_t \leq b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ for all i; otherwise, set $x_t = 0$.

Do both Learning Mechanisms for the CP model.

Project-III: First-Order Methods and Value-Iteration for MDP

MDP problem with m states and total n actions:

$$\begin{array}{lccccc}
\min_{\mathbf{x}} & \sum_{j\in\mathcal{A}_1} c_j x_j + & \ldots & + \sum_{j\in\mathcal{A}_m} c_j x_j & & \\
\text{s.t.} & \sum_{j\in\mathcal{A}_1} (\mathbf{e}_1 - \gamma \mathbf{p}_j) x_j + & \ldots & + \sum_{j\in\mathcal{A}_m} (\mathbf{e}_m - \gamma \mathbf{p}_j) x_j & = & \mathbf{e}, \\
 & \ldots & x_j & \ldots & \geq & 0,\ \forall j,
\end{array} \tag{7}$$

$$\begin{array}{lccl}
\text{maximize}_{\mathbf{y}} & \sum_{i=1}^m y_i & & \\
\text{subject to} & y_i - \gamma \mathbf{p}_j^T \mathbf{y} & \leq & c_j,\ \forall j \in \mathcal{A}_i,\ \forall i.
\end{array}$$

where y_i represents the cost-to-go value in state i.

Question 1: Prove that in (7) every basic feasible solution represent a policy, i.e., the basic variables have exactly one variable from each state i. Furthermore, prove each basic variable value is no less than 1, and the sum of all basic variable values is $\frac{m}{1-\gamma}$.

P-III: First-Order Methods and Value-Iteration for MDP

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

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

Question 2: Prove the contraction result:

$$\|\mathbf{y}^{k+1} - \mathbf{y}^*\|_\infty \leq \gamma \|\mathbf{y}^k - \mathbf{y}^*\|_\infty, \ \forall k.$$

where $\mathbf{y}^*$ is the fixed-point or optimal value vector, that is, $y_i^* = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*\}, \ \forall i.$

Question 3: In the VI method, if starting with any vector $\mathbf{y}^0 \geq \mathbf{y}^*$ and assuming $\mathbf{y}^1 \leq \mathbf{y}^0$, then prove the following entry-wise monotone property: $\mathbf{y}^* \leq \mathbf{y}^{k+1} \leq \mathbf{y}^k, \ \forall k.$

On the other hand, if we start from a vector such that $y_i^0 < \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^0\}, \ \forall i$ ($\mathbf{y}^0$ in the interior of the feasible region), then prove the entry-wise monotone property: $\mathbf{y}^* \geq \mathbf{y}^{k+1} \geq \mathbf{y}^k, \ \forall k.$

P-III: Randomized VI

Rather than go through all state values in each iteration, we modify the VI method, call it RamdomVI: In the kth iteration, randomly select a subset of states B^k and do

$$y_i^{k+1} = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^k\}, \ \forall i \in B^k. \tag{8}$$

In RandomVI, we only update a subset of state values at random in each iteration.

Question 4: What can you tell the convergence of the RandomVI method? Does it make a difference with the classical VI method? How is the sample size affect the performance? Use simulated computational experiments to verify your claims.

Importance Sampling: Rather than randomly select a subset of all states in each iteration, suppose we build an "influence tree" from a given subset of states, say B, for all sates, denoted by $I(B)$, that are connected by any state in B. Then when states in B are updated in the current iteration, then selected a subset of states in $I(B)$ for updating in the next iteration. Redo the computational experiments using this strategy for a sparsely connected ($\mathbf{p}_j$ is a very sparse distribution vector for each action j) MDP network. In doing so, many unimportant or irrelevant states may be avoided which results a state-reduction.

P-III: Cyclic VI

Question 5: Here is another modification, called CyclicVI: In the kth iteration do

- Initialize $\tilde{\mathbf{y}}^k = \mathbf{y}^k$.
- For $i = 1$ to m

$$\tilde{y}_i^k = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k\} \tag{9}$$

- $\mathbf{y}^{k+1} = \tilde{\mathbf{y}}^k$.

In the CyclicVI method, as soon as a state value is updated, we use it to update the rest of state values.

What can you tell the convergence of the CyclicVI method? Does it make a difference with other VI methods? Use simulated computational experiments to verify your claims. How is this cyclic method related to the method at the bottom of Question 4?

P-III: Randomly Permuted Cyclic VI

In the CyclicVI method, rather than with the fixed cycle order from 1 to m, we follow a random permutation order, or sample without replacement to update the state values. More precisely, in the kth iteration do

0. Initialize $\tilde{\mathbf{y}}^k = \mathbf{y}^k$ and $B^k = \{1, 2, ..., m\}$

1. – Randomly select $i \in B^k$

   –
$$\tilde{y}_i^k = \min_{j \in \mathcal{A}_i} \{c_j + \gamma \mathbf{p}_j^T \tilde{\mathbf{y}}^k\} \tag{10}$$

   – remove i from B^k and return to Step 1.

3. $\mathbf{y}^{k+1} = \tilde{\mathbf{y}}^k$.

We call it the randomly permuted CyclicVI or RPCyclicVI in short

What can you tell the convergence of the RPCyclicVI method? Does it compare with other VI methods? Use simulated computational experiments to verify your claims.

Project-IV: ADMM

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}. \end{array} \tag{11}$$

We now reformulate the QP problem as

$$\begin{array}{ll} \text{minimize}_{\mathbf{x},\mathbf{x}'} & \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b}, \quad (\mathbf{y}) \\ & \mathbf{x} - \mathbf{x}' = \mathbf{0}; \quad (\mathbf{s}) \\ & \mathbf{x}' \geq \mathbf{0}, \end{array}$$

and consider the split augmented Lagrangian function:

$$L(\mathbf{x}, \mathbf{x}' \geq \mathbf{0}, \mathbf{y}, \mathbf{s}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{c}^T\mathbf{x} - \mathbf{y}^T(A\mathbf{x} - \mathbf{b}) - \mathbf{s}^T(\mathbf{x} - \mathbf{x}') + \frac{\beta}{2}\left(\|A\mathbf{x} - \mathbf{b}\|^2 + \|\mathbf{x} - \mathbf{x}'\|^2\right).$$

P-IV: Questions

- Split constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x}' \geq \mathbf{0}$ and add $\mathbf{x} - \mathbf{x}' = \mathbf{0}$ and apply Two-Block ADMM.
- Further partition $\mathbf{x}$ into p blocks, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p]$ and apply the cyclic multi-block ADMM.
- Randomly permute the update order of $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_p$, followed by update of $\mathbf{x}'$ and then the multipliers $\mathbf{y}, \mathbf{s}$, in each cycle of ADMM.
- In each cycle of ADMM, we randomly assemble variables into $\mathbf{x}_1, ..., \mathbf{x}_p$, and update $\mathbf{x}$ according in the order of 1 to p, followed by updates of $\mathbf{x}'$ and then the multipliers $\mathbf{y}, \mathbf{s}$.
- Consider the case of each $x_j = \{0, 1\}$ and apply RAC-ADMM. Applications include QAP, Max-Bisection, Sparse-Portfolio Selection, etc.