Randomized Block Coordinate and Stochastic (Sub-)Gradient Methods

Yinyu Ye

Stanford University, MS&E and ICME

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

(Chapter 8)

Block Coordinate Descent Method for Unconstrained Optimization I

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

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 $\mathbf{d}$

$$\|\nabla_j f(\mathbf{x} + \mathbf{e}_j.*\mathbf{d}) - \nabla_j f(\mathbf{x})\| \le \beta_j \|\mathbf{e}_j.*\mathbf{d}\| \tag{1}$$

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):

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

Aitken Double Sweep Method:

$$\mathbf{x}_1 \longleftarrow \arg\min_{\mathbf{x}_1} f(\mathbf{x}_1, \ldots, \mathbf{x}_n),$$
$$\vdots$$
$$\mathbf{x}_n \longleftarrow \arg\min_{\mathbf{x}_n} f(\mathbf{x}_1, \ldots, \mathbf{x}_n),$$
$$\mathbf{x}_{n-1} \longleftarrow \arg\min_{\mathbf{x}_{n-1}} f(\mathbf{x}_1, \ldots, \mathbf{x}_n),$$
$$\vdots$$
$$\mathbf{x}_2 \longleftarrow \arg\min_{\mathbf{x}_2} f(\mathbf{x}_1, \ldots, \mathbf{x}_n).$$

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

Randomized Block Coordinate Descent (RBCD) Method:

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

Convergence 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 1 *Let $f : R^n \to R$ be given. For some given point $x^0 \in R^n$, let the level set*

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

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

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

Worst-Case Convergnece Comparison of BCDs

There is a convex quadratic minimization problem of dimension n:

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

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

- CBCD is $\frac{n}{2\pi^2}$ times slower than SDM;
- CBCD is $\frac{n^2}{2\pi^2}$ times slower than RBCD (each iteratione 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 2 *(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 $\|\mathbf{x}\|_\beta^2 = \sum_j \beta_j x_j^2$.

Proof: Denote by $\delta^k = f(\mathbf{x}^k) - f(\mathbf{x}^*)$, $\Delta^k = \mathbf{x}^k - \mathbf{x}^*$, and

$$(r^k)^2 = \|\mathbf{x}^k - \mathbf{x}^*\|_\beta^2 = \sum_j \beta_j (x_j^k - x_j^*)^2.$$

Then, from the RBCGD iteration

$$(r^{k+1})^2 = (r^k)^2 - 2\nabla_{i^k} f(\mathbf{x}^k)(x_{i^k}^k - x_{i^k}^*) + \frac{1}{\beta_{i^k}}(\nabla_{i^k} f(\mathbf{x}^k))^2.$$

It follows from the β-Coordinate Lipschitz condition,

$$\begin{aligned} f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k) &\le \nabla_{i^k} f(\mathbf{x}^k)(x_{i^k}^{k+1} - x_{i^k}^k) + \tfrac{1}{2\beta_{i^k}}(\nabla_{i^k} f(\mathbf{x}^k))^2 \\ &= \tfrac{-1}{2\beta_{i^k}}(\nabla_{i^k} f(\mathbf{x}^k))^2. \end{aligned}$$

Combining the two inequalities, we have

$$(r^{k+1})^2 \le (r^k)^2 - 2\nabla_{i^k} f(\mathbf{x}^k)(x_{i^k}^k - x_{i^k}^*) + 2(f(\mathbf{x}^k) - f(\mathbf{x}^{k+1})).$$

Dividing both sides by 2 and taking expectation with respect to i^k yields

$$E_{i^k}[\frac{1}{2}(r^{k+1})^2] \le \frac{1}{2}(r^k)^2 - \frac{1}{n}\nabla f(\mathbf{x}^k)^T(\mathbf{x}^k - \mathbf{x}^*) + f(\mathbf{x}^k) - E_{i^k}[f(\mathbf{x}^{k+1})],$$

which together with convexity assumption $\nabla f(\mathbf{x}^k)^T(\mathbf{x}^* - \mathbf{x}^k) \leq f(\mathbf{x}^*) - f(\mathbf{x}^k)$ gives

$$E_{i^k}[\frac{1}{2}(r^{k+1})^2] \leq \frac{1}{2}(r^k)^2 + \frac{1}{n}f(\mathbf{x}^*) + \frac{n-1}{n}f(\mathbf{x}^k) - E_{i^k}[f(\mathbf{x}^{k+1})],$$

Rearranging gives, for each $k \geq 0$,

$$E_{i^k}[\frac{1}{2}(r^{k+1})^2 + \delta^{k+1}] \leq \left(\frac{1}{2}(r^k)^2 + \delta^k\right) - \frac{1}{n}\delta^k.$$

Taking expectation with respect to ξ^{k-1} on both sides

$$\begin{aligned} E_{\xi^k}[\tfrac{1}{2}(r^{k+1})^2 + \delta^{k+1}] &\leq E_{\xi^{k-1}}[\tfrac{1}{2}(r^k)^2 + \delta^k] - \tfrac{1}{n}E_{\xi^{k-1}}[\delta^k] \\ &= E_{\xi^k}[\tfrac{1}{2}(r^k)^2 + \delta^k] - \tfrac{1}{n}E_{\xi^k}[\delta^k]. \end{aligned}$$

Recursively applying the inequalities from and noting that $E_{\xi^k}[f(\mathbf{x}^{k+1})]$ is monotonically decreasing

$$\begin{aligned} E_{\xi^k}[\delta^{k+1}] &\leq E_{\xi^k}[\tfrac{1}{2}(r^{k+1})^2 + \delta^{k+1}] \\ &\leq (\tfrac{1}{2}(r^0)^2 + \delta^0) - \tfrac{1}{n}\textstyle\sum_{j=0}^k E_{\xi^k}[\delta^j] \\ &\leq (\tfrac{1}{2}(r^0)^2 + \delta^0) - \tfrac{k+1}{n}E_{\xi^k}[\delta^{k+1}] \end{aligned}$$

which leads to the desired result.

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".

$$\begin{array}{lll} (OPT) & \min_{\mathbf{x}} & f(\mathbf{x},\omega) \\ & \text{s.t.} & \mathbf{c}(\mathbf{x},\omega) \in K \subset R^m. \end{array} \tag{2}$$

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:

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

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.

Information Complexity and Sample Size in SAA

- In SAA, the required number of samples, M, should be larger than the dimension of decision vector and should grow polynomially with the increase of dimensionality. In specific, let $\mathbf{x}^{SAA}$ be the optimal solution from the SAA method. Then to ensure probability

$$P[F(\mathbf{x}^{SAA}) - F(\mathbf{x}^*) \leq \epsilon] \geq 1 - \alpha,$$
$$M = O(\frac{1}{\epsilon^2})(n \ln(\frac{1}{\epsilon}) + \ln(\frac{1}{\alpha})).$$

- If $\mathbf{x}^*$ is sparse or it can be approximated by a sparse solution with cardinality $p << n$, then by adding a regulative penalty function into the objective

$$\min_{\mathbf{x}} \quad \frac{1}{M}\sum_{i=1}^{M} f(\mathbf{x}, \omega^i) + P(\mathbf{x}),$$

the sample size can be reduced to

$$M = O(\frac{1}{\epsilon^2})(\frac{p}{\epsilon}\ln^{1.5}(\frac{n}{\epsilon}) + \ln(\frac{1}{\alpha})); \quad \text{or in convex case: } M = O(\frac{1}{\epsilon^2})(p \ln(\frac{n}{\epsilon}) + \ln(\frac{1}{\alpha})).$$

SGD and its Advantages

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

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

- Works 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.

Variance Reduction in Stochastic Algorithm Design

- The VR technique has been used extensively in the design of fast stochastic methods for solving large–scale optimization problems in machine learning.
- High Level Idea: Reduce the variance of an estimate X by using another estimate Y with known expectation.
- Specifically, consider $Z_\alpha = \alpha(X - Y) + \mathsf{E}[Y]$.
  - $\mathsf{E}[Z_\alpha] = \alpha \cdot \mathsf{E}[X] + (1 - \alpha) \cdot \mathsf{E}[Y]$
  - $\mathrm{var}(Z_\alpha) = \mathsf{E}\left[(Z_\alpha - \mathsf{E}[Z_\alpha])^2\right] = \alpha^2 \left[\mathrm{var}(X) + \mathrm{var}(Y) - 2\mathrm{cov}(X, Y)\right]$
  - When $\alpha = 1$, we have $\mathsf{E}[Z_\alpha] = \mathsf{E}[X]$, which is useful for establishing concentration bounds.
  - When $\alpha < 1$, Z_α will potentially have a smaller variance than X, but we no longer have $\mathsf{E}[Z_\alpha] = \mathsf{E}[X]$. (In what follows, we let $\alpha = 1$.)
  - Overall, variance reduction occur if $\mathrm{cov}(X, Y) > 0$.

VR Illustration: Finite–Sum Minimization I

- Consider the following so–called finite–sum minimization problem:

$$\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x}) \right\}. \tag{3}$$

  Here, $f_1, \ldots, f_M$ are smooth (convex) loss functions and M is huge so that the computation of $\nabla F(\cdot)$ is costly.

- Examples
  - Linear regression: $f_i(\mathbf{x}) = (\mathbf{a}_i^T \mathbf{x} - b_i)^2$
  - Logistic regression: $f_i(\mathbf{x}) = \ln\left(1 + \exp\left(b_i \mathbf{a}_i^T \mathbf{x}\right)\right)$

- Stochastic Gradient Descent (SGD): choose i_k from $\{1, ..., M\}$ uniformly at random and let

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

  - We have $\mathsf{E}\left[\nabla f_{i_k}(\mathbf{x}^k)\right] = \nabla F(\mathbf{x}^k)$, but variance of the estimate can be large.
  - To guarantee convergence, we generally need diminishing step sizes (e.g., $\alpha_k = O(k^{-1})$).

VR Illustration: Finite–Sum Minimization II

- Now let $X = \nabla f_{i_k}(\mathbf{x}^k)$ for estimating $\nabla F(\mathbf{x}^k)$. What Y should we use to reduce the variance of the estimate?
  - Try $Y = \nabla f_{i_k}(\tilde{\mathbf{x}}^k)$ for some fixed $\tilde{\mathbf{x}}^k$.
  - Note that $\mathsf{E}[Y] = \nabla F(\tilde{\mathbf{x}}^k)$.
- Now, form $Z = X - Y + \mathsf{E}[Y] = \nabla f_{i_k}(\mathbf{x}^k) - \nabla f_{i_k}(\tilde{\mathbf{x}}^k) + \nabla F(\tilde{\mathbf{x}}^k)$ and set

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \alpha_k \left(\nabla f_{i_k}(\mathbf{x}^k) - \nabla f_{i_k}(\tilde{\mathbf{x}}^k) + \nabla F(\tilde{\mathbf{x}}^k) \right).$$

- Since the computation of $\nabla F(\tilde{\mathbf{x}}^k)$ is costly, we don't want to update $\tilde{\mathbf{x}}^k$ too often but only once for a while.
- This is the core idea behind the stochastic variance–reduced gradient (SVRG) method, which has generated much recent research; see *Accelerating Stochastic Gradient Descent Using Predictive Variance Reduction*, NIPS 2013.

VR Illustration: Finite–Sum Minimization III

- One choice is to update $\tilde{\mathbf{x}}$ at a uniform (or geometric) pace, that is, when $k = rK$ (or $k = 2^r$) for a nonnegative integer r, we let $\tilde{\mathbf{x}}^k = \mathbf{x}^k$ and it remains unchanged from iteration k to $k + K$ (or $2k$).
- Thus, from iteration 1 to k, $\tilde{\mathbf{x}}^k$ is updated, or $\nabla F(\tilde{\mathbf{x}}^k)$ is computed, only k/K (or $\log(k)$) times.
- Moreover, most likely $\mathrm{cov}(\mathbf{x}^k, \tilde{\mathbf{x}}^k) > 0$ during the iteration period k to $k + K$, since both $\mathbf{x}^k$ and $\tilde{\mathbf{x}}^k$ converge to the same limit solution.

VR Illustration: Finite–Sum Minimization IV

- The VR-SGD method can be shown to converge linearly when F satisfies the so–called error bound condition: there exists a $\tau > 0$ such that

$$\mathrm{dist}(\mathbf{x}, \mathcal{X}^*) \leq \tau \|\nabla F(\mathbf{x})\|_2 \quad \text{for all } \mathbf{x}, \tag{4}$$

  where $\mathcal{X}^*$ is the set of optimal solutions.

- If F is strongly convex, then it satisfies 4. However, the converse need not hold; for details, see *Non-Asymptotic Convergence Analysis of Inexact Gradient Methods for Machine Learning Without Strong Convexity*. Optim. Methods Softw. 32(4): 963–992, 2017.

- Extensions of the VR-SGD method to the case where F is non–convex have been proposed and analyzed in *Stochastic Variance Reduction for Nonconvex Optimization*. ICML 2016, and *Variance Reduction for Faster Nonconvex Optimization*. ICML 2016.

Case 1: 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 Hoeffdings 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 2: Online Linear Programming (OLP) Problem

At time $t = 1, ..., n$,

$$r_1 x_1 + \cdots + r_t x_t + \cdots ? \; \cdots$$

$$\begin{pmatrix} | & & | & | & & & & \\ \mathbf{a}_1 & \cdots & \mathbf{a}_t & ? & \cdots & \cdots & ? \\ | & & | & | & & & & \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_t \\ ? \\ \vdots \\ ? \end{pmatrix} \leq \mathbf{b}$$

Decision: $x_t \in [0, 1]$

Previous decisions already made: $x_1, \cdots, x_{t-1}$

Algorithm Motivation from the Offline Primal&Dual LPs

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:

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

Equivalently, by removing s_j's,

$$\begin{array}{ll} \min & \mathbf{b}^\top \mathbf{p} + \sum_{j=1}^n \left(r_j - \mathbf{a}_j^\top \mathbf{p}\right)^+ \\ \text{s.t.} & \mathbf{p} \ge 0 \end{array}$$

$(\cdot)^+$ is the positive-part or ReLu function.

Equivalent Form of the Dual Problem (II)

Normalize the objective, the large-sum functions become SAA:

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

We know

- The primal optimal solution is largely determined by the dual optimal $\mathbf{p}_n^*$
- $\mathbf{p}_n^*$ is the optimal solution of the above sample average approximation

Implication for online LP when orders coming randomly:

- At time t, one can solve $f_t(\mathbf{p})$ (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" and/or "boosting" 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.