(Sub-)Gradient and Stochastic Gradient

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

`http://www.stanford.edu/˜yyye`

(LY: Chapters 8.8)

The (Sub-)Gradient Method

To minimize a continuous function $f(\mathbf{x}),\ \mathbf{x} \in R^n$, at the kth step, the method basically computes

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

which is to go along the steepest descent direction with the step-size $\frac{1}{\beta^k}$, if $f(\mathbf{x})$ is differentiable.

Here $\nabla f(\mathbf{x}^k)$ could be a sub-gradient at $\mathbf{x}^k$, that is, a vector $\boldsymbol{g}$ such that

$$f(\mathbf{x}) - f(\mathbf{x}^k) \geq \boldsymbol{g}^T(\mathbf{x} - \mathbf{x}^k),\ \forall \mathbf{x}$$

when $f(\mathbf{x})$ is a convex function. The key is to choose the step-size

- Fixed Step-Size: $\beta^k = \beta$ where β is a Lipschitz constant.
- Diminished Step-Size: $\beta^k = k + 1$ or $\beta^k = \sqrt{k+1}$

Typically, the method converges in $O(\epsilon^{-2})$ steps to compute and ϵ-KKT solution.

The Stochastic (Sub-)Gradient Method

To minimize a continuous function with the form

$$f(\mathbf{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{x}), \ \mathbf{x} \in R^n.$$

At the kth step, the method basically computes

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{1}{\beta^k}\nabla f_i(\mathbf{x}^k)$$

where i is uniformly and randomly chosen.

There are different way to implement the method

- Sample with replacement till “converges”
- Cyclic: in each cycle do sample without replacement

(Projects #3 on LP)

Online Linear Programming (OLP) Problem

Recall that in Lecture 5, we define the OLP problem

$$\begin{array}{ll} \max & \sum_{j=1}^{n} r_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \le b_i, \;\; i = 1, ..., m \\ & 0 \le x_j \le 1, \;\; j = 1, ..., n \end{array}$$

where $r_j \in \mathbb{R}$, $\mathbf{a}_j = \{a_{ij}\}_{i=1}^m \in \mathbb{R}^m$, and $\mathbf{b} = \{b_i\}_{i=1}^m \in \mathbb{R}_+^m$.

- $(r_j, \mathbf{a}_j)$ are revealed sequentially
- Decide x_j's sequentially
- Conform to constraints

Online Linear Programming (OLP) Problem

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

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

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

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

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

Motivate the Algorithm 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 $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{p} \in \mathbb{R}^m$, $\mathbf{s} \in \mathbb{R}^n$

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

LP duality 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 problem

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

Equivalently, by removing s_j's,

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

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

Equivalent Form of the Dual Problem (II)

Normalize the objective,

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

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 price vector.

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

```
1: Input: $\mathbf{d} = \mathbf{b}/n$
2: Initialize $\mathbf{p}_1 = \mathbf{0}$
3: For $t = 1, 2, ..., n$
4:
      $$x_t = \begin{cases} 1, & \text{if } r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{if } r_t \le \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$$
5: Compute
      $$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \alpha_t \left(\mathbf{a}_t x_t - \mathbf{d}\right) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$
6: 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”.

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

Remarks:

- 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.

The algorithm works for both the stochastic input model and the random permutation model. In the following, we will analyze the algorithm performance under these two models separately.

Model and Data Assumptions

Stochastic Input Model (i.i.d.):

(a) The column-coefficient pair $(\pi_j, \mathbf{a}_j)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$.

Random Permutation Model:

(a') The column-coefficient pair $(\pi_j, \mathbf{a}_j)$'s are (adversarially) determined in advance but arrive in a random order (sample without replacement).

Both assume boundedness:

(b) There exist constants $\bar{\pi}$ and $\bar{a}$ such that $|\pi_j| \leq \bar{\pi}$ and $\|\mathbf{a}_j\|_\infty \leq \bar{a}$ for $j = 1, ..., n$.

(c) The right-hand-side $\mathbf{b} = n\mathbf{d}$ and there exists positive constant $\underline{d}$ and $\bar{d}$ such that $\underline{d} \leq d_i \leq \bar{d}$ for $i = 1, ..., m$.

Let

- Ξ denote the family of distributions that satisfy (b)
- $\Xi_{\mathcal{D}}$ denote all the data sets $\mathcal{D} = \{(\pi_j, \mathbf{a}_j)\}_{j=1}^n$ that satisfy (b).

Performance Metrics (I)

Let "Offline" optimal solution be $\mathbf{x}^* = (x_1^*, ..., x_n^*)$ and "online" solution be $\mathbf{x} = (x_1, ..., x_n)$.

$$R_n^* = \sum_{j=1}^{n} \pi_j x_j^*$$

$$R_n = \sum_{j=1}^{n} \pi_j x_j .$$

Objective: Minimize the worst-case gap/regret between the offline and online objective values

$$\begin{aligned} \Delta_n &= \sup_{\mathcal{P}\in\Xi} \mathbb{E}_{\mathcal{P}}\left[R_n^* - R_n\right] \quad \text{(Stochastic Input)} \\ \delta_n &= \sup_{\mathcal{D}\in\Xi_{\mathcal{D}}} R_n^* - \mathbb{E}\left[R_n\right] \quad \text{(Random Permutation)} \end{aligned}$$

where the expectation in the first line is taken with respect to the distribution $\mathcal{P}$ that generates $(\pi_j, \mathbf{a}_j)$'s, and the expectation in the second line is taken with respect to the random permutation ordering of $(r_j, \mathbf{a}_j)$'s. Also Note that $R_n^* \geq O(n)$.

Performance Metrics (II)

Also, we measure the possible constraint violation of the online solution

$$v(\mathbf{x}) = \| (A\mathbf{x} - \mathbf{b})^+ \|_2$$

Remark: A bi-objective performance measure

Different from *Online Convex Optimization*:

- Presence of the constraints
- A stronger benchmark: the “offline” optimal $x_1^*, ..., x_n^*$ are allowed to take different values in online LP but are required to be the same for OCO

Understanding $f_n(\mathbf{p})$ in the i.i.d Model

Under the stochastic input model, when the column-coefficient pair $(r_j, \mathbf{a}_j)$'s are i.i.d. sampled from an unknown distribution $\mathcal{P}$, define the stochastic program

$$\begin{array}{ll} \min\ f(\mathbf{p}) := & \mathbf{d}^\top \mathbf{p} + \mathbb{E}_{(r,\mathbf{a})\sim\mathcal{P}}\left[(r - \mathbf{a}^\top \mathbf{p})^+\right] \\ \text{subject to} & \mathbf{p} \geq \mathbf{0}, \end{array}$$

Observation: $f_n(\mathbf{p})$ is a sample average approximation for $f(\mathbf{p})$

Denote $\mathbf{p}^*$ as the optimal solution of $f(\mathbf{p})$

Convergence of $\mathbf{p}_n$: Sample Size Theorem

Theorem 1 (Dual Convergence) *Under moderate conditions that guarantees a local strong convexity of $f(\mathbf{p})$ around $\mathbf{p}^*$, there exists a constant C such that*

$$\mathbb{E}\|\mathbf{p}_n^* - \mathbf{p}^*\|_2^2 \leq \frac{Cm \log\log n}{n}$$

holds for all $n > m$ and $\mathcal{P} \in \Xi$.

L$_2$ convergence for the dual optimal solution is stronger than other types of convergence.

Heuristically,

$$\mathbf{p}_n^* \approx \mathbf{p}^* + \frac{1}{\sqrt{n}} \cdot \textbf{Noise}$$

Boundedness of $\mathbf{p}_t$ under SO Algorithm

Lemma 1 *If the step size* $\alpha_t \leq 1$ *for* $t = 1, ..., n$ *in SO algorithm, then*

$$\|\mathbf{p}_t\|_2 \leq \frac{2\bar{r} + m(\bar{a} + \bar{d})^2}{\underline{d}} + m(\bar{a} + \bar{d})$$

with probability 1 *for* $t = 1, ..., n$, *where* $\mathbf{p}_t$*'s are specified by SO algorithm.*

Remarks:

- The boundedness is crucial for the algorithm analysis
- It is automatically guaranteed by the algorithm. In other words, no explicit projection (which could be computationally costly) to certain bounded region is required for the algorithm.

Performance Analysis for Stochastic Input Model

Theorem 2 *With step size $\alpha_t = 1/\sqrt{n}$, the regret and expected constraint violation of the algorithm satisfy*

$$\mathbb{E}[R_n^* - R_n] \le m(\bar{a} + \bar{d})^2\sqrt{n}$$

$$\mathbb{E}\left[v(\mathbf{x})\right] \le \left(\frac{2\bar{r} + m(\bar{a}+\bar{d})^2}{\underline{d}} + m(\bar{a}+\bar{d})\right)\sqrt{n}.$$

hold for all $m, n \in \mathbb{N}^+$ and distribution $\mathcal{P} \in \Xi$.

Remark:

- Both are on the order of $m\sqrt{n}$

Proof Sketch

For the proof of the **regret**, it utilizes the structure of the LP and largely mimics the analyses of the stochastic/online gradient descent algorithm

For the proof of the **constraint violation**, the updating equation tells

$$\mathbf{p}_{t+1} \geq \mathbf{p}_t + \frac{1}{\sqrt{n}}(\mathbf{a}_t x_t - \mathbf{d})$$

where the inequality is elementwise. Therefore,

$$\begin{aligned}\textstyle\sum_{t=1}^n \mathbf{a}_t x_t \quad &\leq n\mathbf{d} + \textstyle\sum_{t=1}^n \sqrt{n}(\mathbf{p}_{t+1} - \mathbf{p}_t)\\ &\leq \mathbf{b} + \sqrt{n}\mathbf{p}_{n+1}\end{aligned}$$

The proof is completed by plugging in the boundedness of $\mathbf{p}_t$'s

Challenge of Random Permutation Model

Why study the random permutation model:

- From the perspective of online LP, the random permutation model can be interpreted as an online algorithm that solves an online LP problem under data generation assumptions that are weaker than the i.i.d. assumptions. Hence, the stochastic input model can be viewed as a special case of the random permutation model.
- From the perspective of solving offline integer LPs, the permutation creates the randomness for integer LPs when there is no inherent randomness with the coefficients.

Challenges in analyzing the random permutation model:

- Under the stochastic input model, $(r_j, \mathbf{a}_j)$'s are i.i.d. drawn from $\mathcal{P}$. In the online context, it is easier to infer the future samples based on the past observations.
- Under the random permutation model, it is more challenging to infer the future samples based on the past observations. Because the values of the entries $(r_j, \mathbf{a}_j)$ can be adversarially chosen in advance.

Useful Ideas

Under the stochastic input model,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] = \frac{1}{n}\mathbb{E}\left[R_n^*\right]$$

Under the random permutation model, heuristically,

$$\frac{1}{t}\mathbb{E}\left[R_t^*\right] \geq \frac{1}{n}R_n^* - O\left(\frac{\log t}{t}\right)$$

The quantity $O\left(\frac{\log t}{t}\right)$ can be interpreted as an information toll for the relaxation of the data generation mechanism

Here expectation is not taken for the offline optimal R_n^* under the random permutation model because it is a deterministic value given the data set $\mathcal{D} = \{(r_j, \mathbf{a}_j)\}_{j=1}^n$

Performance Analysis for Random Permutation Model

Theorem 3 *With the fixed step size* $\alpha_t = \frac{1}{\sqrt{n}}$ *for* $t = 1, ..., n$, *the regret and expected constraint violation of the SO algorithm satisfy*

$$R_n^* - \mathbb{E}[R_n] \leq O\left((m + \log n)\sqrt{n}\right)$$

$$\mathbb{E}\left[v(\mathbf{x})\right] \leq O(m\sqrt{n}).$$

for all $m, n \in \mathbb{N}^+$ *and* $\mathcal{D} \in \Xi_D$.

Remark:

- Extra term $O(\sqrt{n}\log n)$ compared to the stochastic input case

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

```
1: Initialize $\mathbf{b}_1 = \mathbf{b}$
2: Initialize $\mathbf{p}_1 = \mathbf{0}$
3: For $t = 1, 2, ..., n$
4:
       $$x_t = \begin{cases} 1, & \text{if } r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & \text{if } r_t \le \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$$
5: Compute
       $$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \alpha_t \left( \mathbf{a}_t x_t - \tfrac{1}{n-t+1} \mathbf{b}_t \right) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$
6: Update remaining inventory: $\mathbf{b}_{t+1} = \mathbf{b}_t - \mathbf{a}_t x_t$.
7: Return $\mathbf{x} = (x_1, ..., x_n)$
```

The average inventory vector is adaptively adjusted based on the previous realizations/decisions – this is a non-stationary approach.

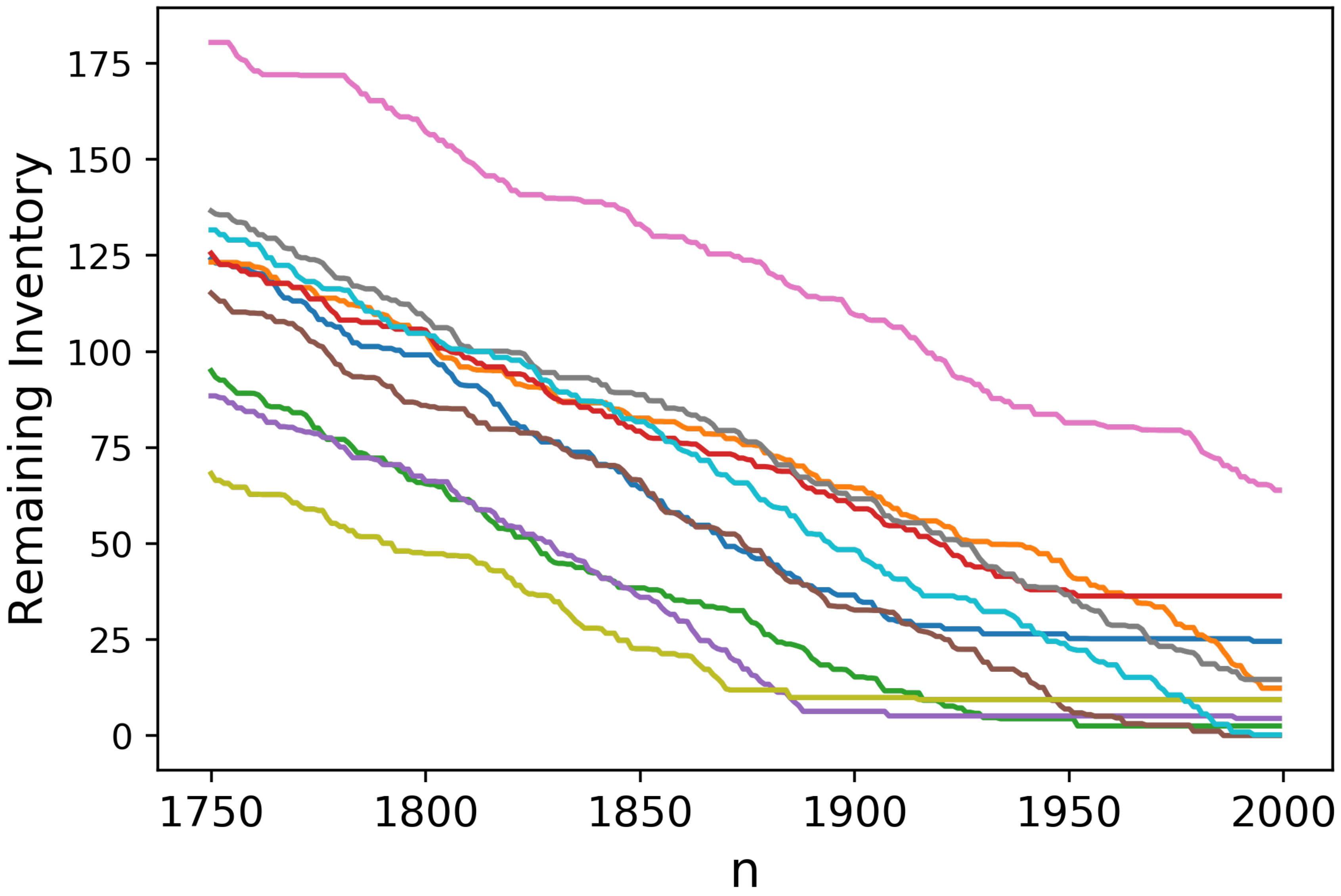

Figure 1: The first resource usage in the Non-Adaptive algorithm, 10 runs

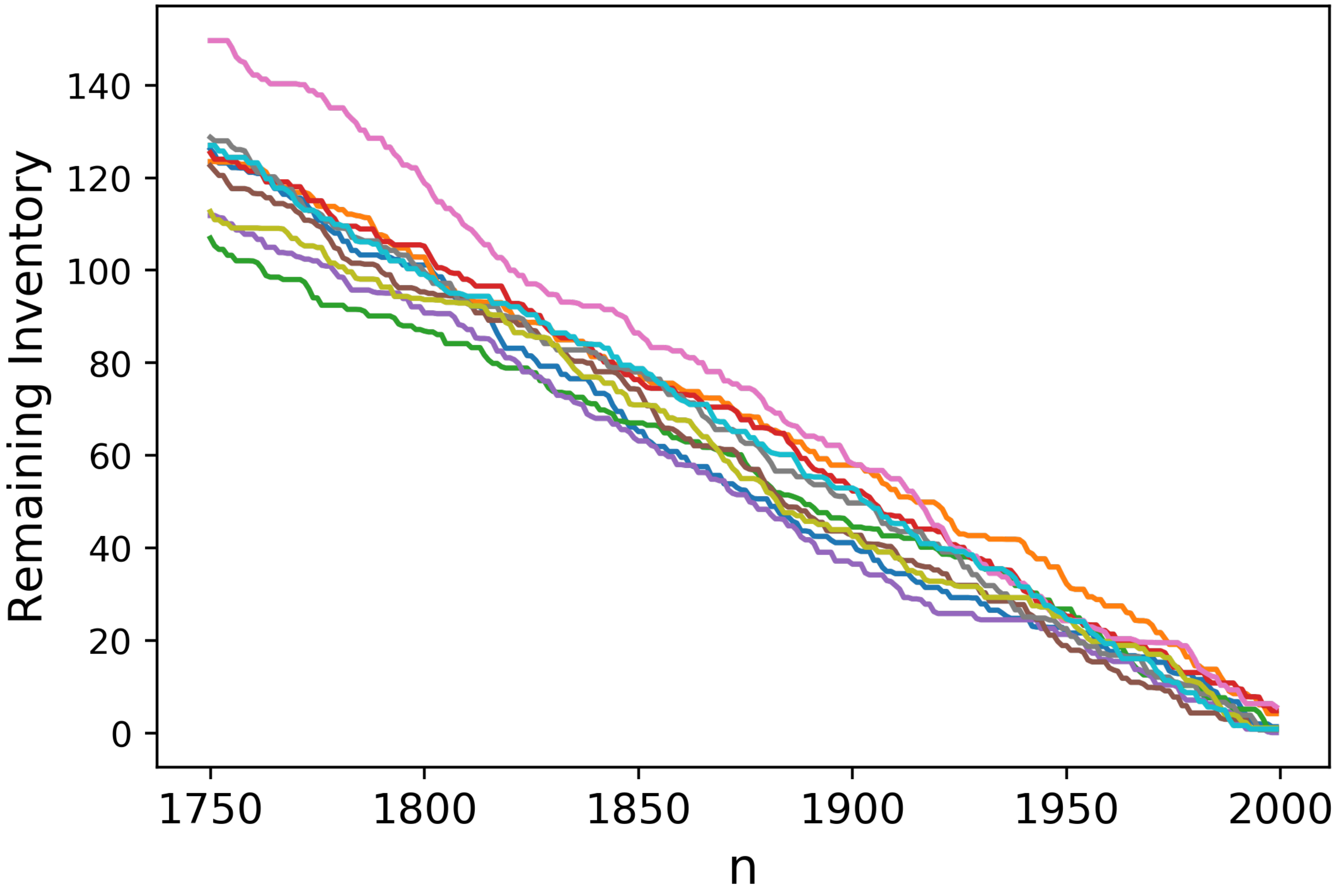

Figure 2: The first resource usage in the Adaptive algorithm, 10 runs

Limitation of the Previous Algorithm/Analyses

- A crucial assumption is that the right-hand-side $\mathbf{b} = n\mathbf{d}$ scales linearly with n
- This assumption makes sense in many but not all application contexts. For some LP problems, one may have $\mathbf{b} = o(n)$ or even $o(1)$
- Is there a remedy for this case? Also, we do not want to compromise the computational efficiency of the SO algorithm

A Boosted Variable-Replicating Algorithm

For solving an Offline LP problem,

$$\begin{array}{ll} \max & \sum_{j=1}^{n} r_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \le b_i, \;\; i = 1, ..., m \\ & 0 \le x_j \le 1, \;\; j = 1, ..., n \end{array}$$

Consider a new LP with replication:

$$\begin{array}{ll} \max & \sum_{j=1}^{n} \sum_{h=1}^{k} r_j x_{jh} \\ \text{s.t.} & \sum_{j=1}^{n} \sum_{h=1}^{k} a_{ij} x_{jh} \le k b_i, \;\; i = 1, ..., m \\ & 0 \le x_j \le 1, \;\; j = 1, ..., n. \end{array}$$

Algorithm: Solve the new LP with SO Algorithm and use $x_j = \frac{1}{k}(x_{j1} + ... + x_{jk})$ as the solution to the original LP.

The replication is symbolic without increase the storage of the data.

A Boosted Variable-Replicating Algorithm

Remarks:

- By replacing $x_j = \frac{1}{k}(x_{j1} + ... + x_{jk})$, we scale up the right-hand-side with a factor of k. Also, the number of variables go up with a factor of k

- Although the solution may no longer be integer-valued, rounding method can provide a binary solution based on the real-valued solution.

- For more general LP that decision variable x_j are between $[0, M_j]$, we can replicate x_j for M_j times for each x_j and apply this algorithm.

Precisely, for solving an Offline LP problem with the following form,

$$\begin{array}{ll} \max & \sum_{j=1}^{n} r_j x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \;\; i = 1, ..., m \\ & x_j \in [0, M_j], \;\; j = 1, ..., n \end{array}$$

We solve the boosted problem with variable-replication:

$$\begin{array}{ll} \max & \sum_{j=1}^{n} \sum_{h=1}^{kM_j} r_j x_{jh} \\ \text{s.t.} & \sum_{j=1}^{n} \sum_{h=1}^{kM_j} a_{ij} x_{jh} \le kb_i, \;\; i = 1, ..., m \\ & 0 \le x_{jh} \le 1, \;\; j = 1, ..., n \end{array}$$

Algorithm: Solve the new LP with SO Algorithm and use $x_j = \frac{1}{k} \sum_{h=1}^{kM_j} x_{jh}$ as the solution to the original LP.

Again, the replication is symbolic without increase the storage of the data.

Multi-knapsack Problem Instances - Binary LP

We generate data as in Chu & Beasley but with $\mathbf{b} = O(\sqrt{n})$ and conduct experiments using C implementation

		V.R. Alg.	Gurobi
$m = 5, n = 500, k = 50$	Time	0.000	0.211
	Cmpt. Ratio	88.2%	95.3%
$m = 5, n = 500, k = 1000$	Time	0.007	0.211
	Cmpt. Ratio	89.2%	95.3%
$m = 8, n = 10^3, k = 50$	Time	0.004	3.800
	Cmpt. Ratio	89.9%	99.0%
$m = 8, n = 10^3, k = 1000$	Time	0.077	3.800
	Cmpt. Ratio	95.6%	99.0%
$m = 64, n = 10^4, k = 50$	Time	0.013	> 60
	Cmpt. Ratio	90.3%	98.7%
$m = 64, n = 10^4, k = 1000$	Time	0.223	> 60
	Cmpt. Ratio	96.4%	98.7%

Multi-knapsack Problem Instances - Binary LP

The same experiment setup but with larger scale

		V.R. Alg.	Gurobi
$m = 32, n = 4000, k = 50$	Time	0.048	> 60
	Cmpt. Ratio	89.4%	95.4%
$m = 32, n = 4000, k = 1000$	Time	0.720	> 60
	Cmpt. Ratio	90.6%	95.4%
$m = 64, n = 10^4, k = 50$	Time	0.392	> 60
	Cmpt. Ratio	89.9%	98.5%
$m = 64, n = 10^4, k = 1000$	Time	3.400	> 60
	Cmpt. Ratio	90.9%	98.5%
$m = 128, n = 10^5, k = 50$	Time	2.100	> 60
	Cmpt. Ratio	91.3%	98.6%
$m = 128, n = 10^5, k = 1000$	Time	45.000	> 60
	Cmpt. Ratio	94.9%	98.6%

More to explore in course Project 3!