Optimization in Data Science and Machine Learning/Decision-Making

AnTai, SJTU, March 17, 2023

Yinyu Ye
Stanford University and CUHKSZ (Sabbatical Leave)
Today’s Talk

1. Online Linear Programming Algorithms and Applications

2. Accelerated Second-Order Methods and Applications

3. Equitable Covering & Partition – Divide and Conquer
1. Online Linear Programming

- 1、在线学习理论与算法研究 (Agrawal et al. 2010, 14, Li&Y 2022)

what is online learning problem?

- traditional machine learning problem: a large amount (training) data, find the best model (examples: regression model, tree model)
  - data: best model

- online learning: data generation and learning are concurrent, influenced by decision (examples: multi-armed bandit problem)
  - data: decision: feedback
    - algorithm: potential model
  - need to learn and optimize simultaneously

From Zizhuo Wang
Linear Programming and LP Giants won Nobel Prize...

\[ \max \sum \pi_j x_j \]

s.t. \[ \sum_j a_j x_j \leq b, \]
\[ 0 \leq x_j \leq 1 \quad \forall \ j = 1, \ldots, n \]

\[ \min \ b^T p + \sum \max \{0, \pi_j - a_j^T p\} \]

s.t. \[ p \geq 0 \]
Online Auction Example

- There is a fixed selling period or number of buyers; and there is a fixed inventory of goods
- Customers come and require a bundle of goods and make a bid
- Decision: To sell or not to sell to each individual customer on the fly?
- Objective: Maximize the revenue.

<table>
<thead>
<tr>
<th>Bid #</th>
<th>$100</th>
<th>$30</th>
<th>....</th>
<th>...</th>
<th>...</th>
<th>Inventory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>x1</td>
<td>x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>1</td>
<td>0</td>
<td>....</td>
<td>...</td>
<td>...</td>
<td>100</td>
</tr>
<tr>
<td>Shoes</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
</tr>
<tr>
<td>T-Shirts</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>500</td>
</tr>
<tr>
<td>Jackets</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>200</td>
</tr>
<tr>
<td>Hats</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>1000</td>
</tr>
</tbody>
</table>
Price Mechanism for Online Auction

- Learn and compute itemized optimal prices
- Use the prices to price each bid
- Accept if it is an over bid, and reject otherwise

<table>
<thead>
<tr>
<th>Bid #</th>
<th>$100</th>
<th>$30</th>
<th>....</th>
<th>...</th>
<th>...</th>
<th>Inventory</th>
<th>Price?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision</td>
<td>x1</td>
<td>x2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>1</td>
<td>0</td>
<td>....</td>
<td>...</td>
<td>...</td>
<td>100</td>
<td>45</td>
</tr>
<tr>
<td>Shoes</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>50</td>
<td>45</td>
</tr>
<tr>
<td>T-Shirts</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>500</td>
<td>10</td>
</tr>
<tr>
<td>Jackets</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td>55</td>
</tr>
<tr>
<td>Hats</td>
<td>1</td>
<td>1</td>
<td>....</td>
<td>...</td>
<td>...</td>
<td>1000</td>
<td>15</td>
</tr>
</tbody>
</table>
Application I: Online Matching for Display Advertising

Jon Stewart Is Retiring, and it's Going to Be (Kind of) Okay

When the news broke Tuesday night that longtime Daily Show host Jon Stewart would be leaving his post in the coming months, the level of trauma on the Internet was palpable. Some expected topics arose, within hours -- minutes, even -- of the announcement trickling out. Why would Stewart leave now? What's his plan? Who should replace him? Could the next Daily Show host be a woman? (Of course.) Is this an elaborate ruse for Stewart to take over the NBC Nightly News? (Of course not.)

The public conversation over the past two days has been so Stewart-centric that the retirement news effectively pushed NBC anchor Brian Williams's suspension off of social media's front pages. Part of that is the shock; we knew the other shoe was about to drop with (on?) Williams, but Stewart's departure was known only to Comedy Central brass before it was revealed to his studio audience. Part of it is how meme-worthy the parallels between the two hosts truly are ("Fake newsmen speak truth, real newsmen spins lies," some post on your Twitter timeline probably read). Breaking at
Revenues generated by different methods

- Total Revenue for impressions in T2 by Greedy and OLP with different allocation risk functions
# of Out-of-Budget Advertisers

- Greedy exhausts budget of many advertisers early.
- Log penalty keeps advertisers in budget but it is very conservative.
- Exponential penalty keeps advertisers in budget until almost the end of the timeframe.
## Detailed Performances

<table>
<thead>
<tr>
<th>Allocation algorithm</th>
<th>Total Revenue</th>
<th>Improvement over greedy</th>
<th>Mid flight oob</th>
<th>Final oob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>$1829.94</td>
<td>-</td>
<td>366</td>
<td>467</td>
</tr>
<tr>
<td>Fixed dual</td>
<td>$1986.67</td>
<td>8.5%</td>
<td>192</td>
<td>452</td>
</tr>
<tr>
<td>Log</td>
<td>$1915.72</td>
<td>4.6%</td>
<td>5</td>
<td>71</td>
</tr>
<tr>
<td>Exponential</td>
<td>$2043.21</td>
<td>11.6%</td>
<td>7</td>
<td>476</td>
</tr>
</tbody>
</table>

oob: out of budget

https://arxiv.org/abs/1407.5710
阿里巴巴在2019年云栖大会上提到在智能履行决策上使用OLP的算法
3.3 MCKP-Allocation

We adopt the primal-dual framework proposed by [2] to solve the problem defined in Equation 5. Let $\alpha$ and $\beta_j$ be the associated dual variables respectively. After obtaining the dual variables, we can solve the problem in an online fashion. Precisely, according to the principle of the primal-dual framework, we have the following allocation rule:

$$x_{ij} = \begin{cases} 
1, & \text{where } j = \arg \max_i (v_{ij} - \alpha c_j) \\
0, & \text{otherwise}
\end{cases} \quad (9)$$
App. II: The Online Algorithm can be Extended to Bandits with Knapsack (BwK)

Applications

• For the previous problem, the decision maker first wait and observe the customer order/arm and then decide whether to accept/play it or not.

• An alternative setting is that the decision maker first decides which order/arm (s)he may accept/play, and then receive a random resource consumption vector $a_j$ and yield a random reward $\pi_j$ of the pulled arm.

• Known as the Bandits with Knapsacks, and it is a tradeoff exploration v.s. exploitation
The decision variable $x_j$ represents the total-times of pulling the j-th arm.

We have developed a two-phase algorithm

- **Phase I**: Distinguish the optimal super-basic variables/arms from the optimal non-basic variables/arms with as fewer number of plays as possible
- **Phase II**: Use the arms in the optimal face to exhaust the resource through an adaptive procedure and achieve fairness

The algorithm achieves a problem dependent regret that bears a logarithmic dependence on the horizon $T$. Also, it identifies a number of LP-related parameters as the bottleneck or condition-numbers for the problem

- Minimum non-zero reduced cost
- Minimum singular-values of the optimal basis matrix.

**First algorithm** to achieve the $O(\log T)$ regret bound [Li, Sun & Y 2021 ICML] (https://proceedings.mlr.press/v139/li21s.html)
min \ f (x), x \in X \in \mathbb{R}^n,

• where \ f \ is nonconvex and twice-differentiable,

\[ g_k = \nabla f(x_k), \quad H_k = \nabla^2 f(x_k) \]

• Goal: find \ x_k \ such that:

\[ \|g_k\| \leq \epsilon \] (primary, first-order condition)

\[ \lambda_{\text{min}}(H_k) \geq -\sqrt{\epsilon} \] (secondary, second-order condition)

• First-order methods typically need \ O(n^2\epsilon^{-2}) \ operations

• Second-order methods typically need \ O(n^3\epsilon^{-1.5}) \ operations

• New? Yes, HSODM: a single-loop method with \ O(n^2\epsilon^{-1.75}) \ operations

**App. III: HSODM for Policy Optimization in Reinforcement Learning**

- Consider policy optimization of linearized objective in reinforcement learning

  \[
  \max_{\theta \in \mathbb{R}^d} L(\theta) := L(\pi_\theta),
  \]

  \[
  \theta_{k+1} = \theta_k + \alpha_k \cdot M_k \nabla \eta(\theta_k),
  \]

- \(M_k\) is usually a preconditioning matrix.

- The Natural Policy Gradient (NPG) method (Kakade, 2001) uses the Fisher information matrix where \(M_k\) is the inverse of

  \[
  F_k(\theta) = \mathbb{E}_{\rho_{\theta_k}, \pi_{\theta_k}} \left[ \nabla \log \pi_{\theta_k}(s, a) \nabla \log \pi_{\theta_k}(s, a)^T \right]
  \]

- Based on KL divergence, TRPO (Schulman et al. 2015) uses KL divergence in the constraint:

  \[
  \max \nabla I_{\rho_\alpha} (\theta_{\alpha})^T (\theta - \theta_{\alpha})
  \]

---

Homogeneous NPG: Apply HSODM!
Dimension Reduced Second-Order Method (DRSOM)

- Motivation from Multi-Directional FOM and Subspace Method, such as CG and ADAM, DRSOM applies the trust-region method in low dimensional subspace.
- This results in a low-dimensional quadratic sub-minimization problem:
- Typically, DRSOM adopts two directions \(d = -\alpha^1 \nabla f(x_k) + \alpha^2 d_k\)
  where \(g_k = \nabla f(x_k), H_k = \nabla^2 f(x^k), d_k = x_k - x_{k-1}\)
- Then we solve a 2-d quadratic minimization problem:

\[
\min m_k^\alpha(\alpha) := f(x_k) + (c_k)^T \alpha + \frac{1}{2} \alpha^T Q_k \alpha
\]

\[
\|\alpha\|_{G_k} \leq \Delta_k
\]

\[
G_k = \begin{bmatrix}
g_k^T g_k & -g_k^T d_k \\
g_k^T d_k & d_k^T d_k
\end{bmatrix},
Q_k = \begin{bmatrix}
g_k^T H_k g_k & -g_k^T H_k d_k \\
-g_k^T H_k d_k & d_k^T H_k d_k
\end{bmatrix},
c_k = \begin{bmatrix}
-\|g_k\|^2 \\
g_k^T d_k
\end{bmatrix}
\]
To use DRSOM in machine learning problems

- We apply the mini-batch strategy to a vanilla DRSOM
- Use Automatic Differentiation to compute gradients
- Train ResNet18/ResNet34 Model with CIFAR 10
- Set Adam with initial learning rate $1e^{-3}$
Preliminary Results: Neural Networks and Deep Learning

Training and test results for ResNet18 with DRSOM and Adam

Pros
- DRSOM has rapid convergence (30 epochs)
- DRSOM needs little tuning

Cons
- DRSOM may over-fit the models
- Running time can benefit from Interpolation
- Single direction DRSOM is also good

Good potential to be a standard optimizer for deep learning!

Training and test results for ResNet34 with DRSOM and Adam (https://arxiv.org/abs/2208.00208)
App. V: Sensor Network Location (SNL)

- **Localization**
  - Given partial pairwise measured distance values
  - Given some anchors’ positions
  - Find locations of all other sensors that fit the measured distance values

This is also called graph realization on a fixed dimension Euclidean space
Mathematical Formulation of Sensor Network Location (SNL)

• Consider Sensor Network Location (SNL)

\( N_x = \{(i, j) : \|x_i - x_j\| = d_{ij} \leq r_d\}, N_a = \{(i, k) : \|x_i - a_k\| = d_{ik} \leq r_d\} \)

where \( r_d \) is a fixed parameter known as the radio range. The SNL problem considers the following QCQP feasibility problem,

\[
\|x_i - x_j\|^2 = d_{ij}^2, \forall (i, j) \in N_x \\
\|x_i - a_k\|^2 = d_{ik}^2, \forall (i, k) \in N_a
\]

• Alternatively, one can solve SNL by the nonconvex nonlinear least square (NLS) problem

\[
\min_X \sum_{(i < j, j) \in N_x} \left( \|x_i - x_j\|^2 - d_{ij}^2 \right)^2 + \sum_{(k, j) \in N_a} \left( \|a_k - x_j\|^2 - d_{kj}^2 \right)^2.
\]
Semidefinite Programming Relaxation

**Step 1: Linearization**

\[ \|x_i - x_j\|^2 = x_i^T x_i - 2x_i^T x_j + x_j^T x_j \]
\[ = Y_{ii} x_i^2 - 2Y_{ij} x_i x_j + Y_{jj} x_j^2 \]

\[ \|a_k - x_j\|^2 = a_k^T a_k - 2a_k^T x_j + x_j^T x_j \]
\[ = Y_{jj} a_k^2 - 2Y_{ij} a_k x_j + Y_{jj} x_j^2 \]

Tighten: \( Y = X^T X, \ X = [x_1, \ldots, x_n] \)

**Step 2: Relax**

\[ Y \succeq X^T X \iff Z = \begin{bmatrix} I & X \\ X^T & Y \end{bmatrix} \succeq PSD \]

This is a conic linear program which is a **convex optimization** problem, but \( O(n^{3.5} \log(\epsilon^{-1})) \)

Biswas and Y 2004, So and Y 2005
Sensor Network Location (SNL) 1

- Graphical results using SDP relaxation (Biswa&Y 2004, SO&Y 2007) to initialize the NLS
- $n = 80$, $m = 5$ (anchors), radio range = 0.5, degree = 25, noise factor = 0.05
- Both Gradient Descent and DRSOM can find good solutions!
Sensor Network Location (SNL) II

- Graphical results without SDP relaxation
- DRSOM can still converge to optimal solutions
Sensor Network Location, Large-Scale Instances I

- Test large SNL instances (terminate at 3,000s and $|g_k| \leq 1e^{-5}$)

- Compare GD, CG, and DRSOM. (GD and CG use Hager-Zhang Linesearch)

| n   | m   | |E|   | CG     | DRSOM  | GD     |
|-----|-----|------|-------|--------|--------|--------|
| 500 | 50  | 2.2e+04 | 1.7e+01 | 1.1e+01 | 2.3e+01 |
| 1000| 80  | 4.6e+04 | 7.3e+01 | 3.9e+01 | 1.8e+02 |
| 2000| 120 | 9.4e+04 | 2.5e+02 | 1.4e+02 | 1.1e+03 |
| 3000| 150 | 1.4e+05 | 6.5e+02 | 1.4e+02 | -       |
| 4000| 400 | 1.8e+05 | 1.3e+03 | 5.0e+02 | -       |
| 6000| 600 | 2.7e+05 | 2.0e+03 | 1.1e+03 | -       |
| 10000|1000 | 4.5e+05 | -       | 2.2e+03 | -       |

Table 2: Running time of CG, DRSOM, and GD on SNL instances of different problem size, $|E|$ denotes the number of QCQP constraints. "-" means the algorithm exceeds 3,000s.

- DRSOM has the best running time (benefits of 2nd order info and interpolation!)
Sensor Network Location, Large-Scale Instances II

- Graphical results with 10,000 nodes and 1000 anchors (no noise) within 3,000 seconds
- GD with Line-search and Hager-Zhang CG both timeout
- DRSOM can converge to $|g_k| \leq 10^{-5}$ in 2,200s
Sensor Network Online Tracking, 2D and 3D
Problem Statement: Divide-Conquer

$n$ points are scattered inside a convex polygon $P$ (in 2D) with $m$ vertices. Does there exist a partition of $P$ into $n$ sub-regions satisfying the following:

- Each sub-region is a convex polygon
- Each sub-region contains one point
- All sub-regions have equal area
In the *Voronoi Diagram*, we satisfy the first two properties (each sub-region is convex and contains one point), but the sub-regions have different areas.
Our Result

Not only such an equitable partition always exists, but also we can find it exactly in running time $O(Nn \log N)$, where $N = m + n$. 
App. VI: Wireless Tower - Resource Allocation
### 基于真实商用网络进行模型优化效果的测试验证验证统计结果：

<table>
<thead>
<tr>
<th></th>
<th>小区数</th>
<th>时段</th>
<th>区域平均负载</th>
<th>区域平均吞吐率 (Mb/S)</th>
<th>高负载小区</th>
<th>高负载小区吞吐率 (Mb/S)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>优化前</strong></td>
<td></td>
<td>中午及晚共6小时</td>
<td>31%</td>
<td>5.3</td>
<td>68%</td>
<td>2.3</td>
</tr>
<tr>
<td><strong>优化后</strong></td>
<td>27</td>
<td>6小时</td>
<td>30%</td>
<td>6.12(提升15%)</td>
<td>66%</td>
<td>2.8(提升22%)</td>
</tr>
<tr>
<td><strong>优化前</strong></td>
<td></td>
<td>晚7时话务高峰</td>
<td>37%</td>
<td>3.9</td>
<td>77%</td>
<td>1.6</td>
</tr>
<tr>
<td><strong>优化后</strong></td>
<td></td>
<td></td>
<td>33%</td>
<td>5.2(提升33%)</td>
<td>68%</td>
<td>2.1(提升32%)</td>
</tr>
</tbody>
</table>

### 统计结果：

- **优化前** 负载：31%
- **优化后** 负载：30%
- **优化前** 吞吐率：5.3 Mb/S
- **优化后** 吞吐率：6.12 Mb/S(提升15%)
- **优化前** 吞吐率：68%
- **优化后** 吞吐率：66%
- **优化前** 吞吐率：2.3 Mb/S
- **优化后** 吞吐率：2.8 Mb/S(提升22%)

- **优化前** 吞吐率：37%
- **优化后** 吞吐率：33%
- **优化前** 吞吐率：3.9 Mb/S
- **优化后** 吞吐率：5.2 Mb/S(提升33%)
- **优化前** 吞吐率：77%
- **优化后** 吞吐率：68%
- **优化前** 吞吐率：1.6 Mb/S
- **优化后** 吞吐率：2.1 Mb/S(提升32%)

### 图表展示：

- 优化前负载
- 优化后负载

### 说明：

- 表格展示了不同时间段的优化前后负载和吞吐率的对比。
- 图表直观展示了优化前和优化后的负载和吞吐率分布情况。
App. VII: Street View Application
Map-Making
Overall Takeaways

It is possible to make online decisions for quantitative decision models with performance guarantees close to that of the offline decision-making with complete information.

Second-Order Derivative information matters and better to integrate FOM and SOM on nonlinear optimization!

Decomposition (Divide and Conquer) helps solving large-scale optimization problems.

• THANK YOU