Distributionally Robust Stochastic and Online Optimization Driven by Data/Samples
Models/Algorithms for Learning and Decision Making

Yinyu Ye

1Department of Management Science and Engineering
Institute of Computational and Mathematical Engineering
Stanford University, Stanford

November 14, 2018
(Joint work with many others ...
Outline

- Introduction to Distributionally Robust Optimization (DRO)
- DRO under Moment, Likelihood and Wasserstein Bounds
- Price of Correlation of High-Dimension Uncertainty
- MDP Value-Iteration Sample/Computation Complexities
- Online Linear Optimization and Dynamic Resource Allocation
Develop *tractable and provable* models and algorithms for optimization with *uncertain and online* data.
We start from considering a **stochastic optimization** problem as follows:

\[
\text{maximize}_{x \in X} \quad \mathbb{E}_{F_\xi}[h(x, \xi)]
\]

where \( x \) is the decision variable with feasible region \( X \), \( \xi \) represents random variables satisfying joint distribution \( F_\xi \).
We start from considering a stochastic optimization problem as follows:

$$\max_{x \in X} \mathbb{E}_{F_\xi}[h(x, \xi)]$$

(1)

where $x$ is the decision variable with feasible region $X$, $\xi$ represents random variables satisfying joint distribution $F_\xi$.

- **Pros:** In many cases, the expected value is a good measure of performance.
- **Cons:** One has to know the exact distribution of $\xi$ to perform the stochastic optimization. Deviant from the assumed distribution may result in sub-optimal solutions. Even know the distribution, the solution/decision is generically risky.
Learning with Noises

“panda”
57.7% confidence

+ \epsilon

= “gibbon”
99.3% confidence
Learning with Noises

\[ \text{“panda”} + \epsilon = \text{“gibbon”} \]

57.7% confidence

99.3% confidence

Goodfellow et al. [2014]
In order to overcome the lack of knowledge on the distribution, people proposed the following (static) robust optimization approach:

\[
\max_{\mathbf{x} \in \mathcal{X}} \min_{\xi \in \Xi} h(\mathbf{x}, \xi)
\]  

(2)

where \( \Xi \) is the support of \( \xi \).
Robust Optimization

In order to overcome the lack of knowledge on the distribution, people proposed the following (static) robust optimization approach:

$$\maximize_{x \in X} \min_{\xi \in \Xi} h(x, \xi)$$  \hspace{1cm} (2)

where $\Xi$ is the support of $\xi$.

- **Pros**: Robust to any distribution; only the support of the parameters are needed.
- **Cons**: Too conservative. The decision that maximizes the worst-case pay-off may perform badly in usual cases; e.g., Ben-Tal and Nemirovski [1998, 2000], etc.
In practice, although the exact distribution of the random variables may not be known, people usually know certain observed samples or training data and other statistical information.
Motivation for a Middle Ground

In practice, although the exact distribution of the random variables may not be known, people usually know certain observed samples or training data and other statistical information.

Thus we could choose an intermediate approach between stochastic optimization, which has no robustness in the error of distribution; and the robust optimization, which admits vast unrealistic single-point distribution on the support set of random variables.
Distributionally Robust Optimization

A solution to the above-mentioned question is to take the following Distributionally Robust Optimization/Learning (DRO) model:

$$\max_{x \in X} \min_{F_\xi \in \mathcal{D}} \mathbb{E}_{F_\xi} [h(x, \xi)]$$  \hspace{1cm} (3)$$

In DRO, we consider a set of distributions $\mathcal{D}$ and choose one to maximize the expected value for any given $x \in X$. 
A solution to the above-mentioned question is to take the following Distributionally Robust Optimization/Learning (DRO) model:

$$\max_{x \in X} \min_{F_\xi \in D} \mathbb{E}_{F_\xi} [h(x, \xi)]$$

In DRO, we consider a set of distributions $D$ and choose one to maximize the expected value for any given $x \in X$.

When choosing $D$, we need to consider the following:

- **Tractability**
- **Practical (Statistical) Meanings**
- **Performance** (the potential loss comparing to the benchmark cases)
Sample History of DRO

- First introduced by Scarf [1958] in the context of inventory control problem with a single random demand variable.
- Distribution set based on moments: Dupacova [1987], Prekopa [1995], Bertsimas and Popescu [2005], Delage and Y [2007,2010], etc
- Distribution set based on Likelihood/Divergences: Nilim and El Ghaoui [2005], Iyanger [2005], Wang, Glynn and Y [2012], etc
- Distribution set based on Wasserstein ambiguity set: Mohajerin Esfahani and Kuhn [2015], Blanchet, Kang, Murthy [2016], Duchi, Glynn, Namkoong [2016]
- Axiomatic motivation for DRO: Delage et al. [2017]; Ambiguous Joint Chance Constraints Under Mean and Dispersion Information: Hanasusanto et al. [2017]
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to Distributionally Robust Optimization</td>
</tr>
<tr>
<td>2</td>
<td>DRO under Moment, Likelihood and Wasserstein Bounds</td>
</tr>
<tr>
<td>3</td>
<td>Price of Correlation of High-Dimension Uncertainty</td>
</tr>
<tr>
<td>4</td>
<td>MDP Value-Iteration Sample/Computation Complexities</td>
</tr>
<tr>
<td>5</td>
<td>Online Linear Optimization and Dynamic Learning</td>
</tr>
</tbody>
</table>
Define

\[ \mathcal{D} = \left\{ F_\xi \left| \begin{array}{l} P(\xi \in \Xi) = 1 \\ (\mathbb{E}[\xi] - \mu_0)^T \Sigma_0^{-1} (\mathbb{E}[\xi] - \mu_0) \leq \gamma_1 \\ \mathbb{E}[(\xi - \mu_0)(\xi - \mu_0)^T] \preceq \gamma_2 \Sigma_0 \end{array} \right. \right\} \]

That is, the distribution set is defined based on the support, first and second order moments constraints.
DRO with Moment Bounds

Define

\[ D = \left\{ F_\xi \mid P(\xi \in \Xi) = 1, \quad (\mathbb{E}[\xi] - \mu_0)^T \Sigma_0^{-1} (\mathbb{E}[\xi] - \mu_0) \leq \gamma_1, \quad \mathbb{E}[(\xi - \mu_0)(\xi - \mu_0)^T] \leq \gamma_2 \Sigma_0 \right\} \]

That is, the distribution set is defined based on the support, first and second order moments constraints.

Theorem

Under mild technical conditions, the DRO model can be solved to any precision \( \epsilon \) in time polynomial in \( \log (1/\epsilon) \) and the sizes of \( \mathbf{x} \) and \( \xi \)

Delage and Y [2010]
Confidence Region on $F_\xi$

Does the construction of $\mathcal{D}$ make a statistical sense?
Confidence Region on $F_{\xi}$

Does the construction of $D$ make a statistical sense?

**Theorem**

Consider

$$D(\gamma_1, \gamma_2) = \left\{ F_{\xi} \left| \begin{array}{l} P(\xi \in \Xi) = 1 \\ (\mathbb{E}[\xi] - \mu_0)^T \Sigma_0^{-1}(\mathbb{E}[\xi] - \mu_0) \leq \gamma_1 \\ \mathbb{E}[(\xi - \mu_0)(\xi - \mu_0)^T] \leq \gamma_2 \Sigma_0 \end{array} \right. \right\}$$

where $\mu_0$ and $\Sigma_0$ are point estimates from the empirical data (of size $m$) and $\Xi$ lies in a ball of radius $R$ such that $||\xi||_2 \leq R$ a.s..

Then for $\gamma_1 = O\left(\frac{R^2}{m} \log (4/\delta)\right)$ and $\gamma_2 = O\left(\frac{R^2}{\sqrt{m}} \sqrt{\log (4/\delta)}\right)$,

$$P(F_{\xi} \in D(\gamma_1, \gamma_2)) \geq 1 - \delta$$
DRO with Likelihood Bounds

Define the distribution set by the constraint on the likelihood ratio. With observed Data: $\xi_1, \xi_2, \ldots, \xi_N$, we define

$$\mathcal{D}_N = \left\{ F_\xi \left| \begin{array}{l} P(\xi \in \Xi) = 1 \\ L(\xi, F_\xi) \geq \gamma \end{array} \right. \right\}$$

where $\gamma$ adjusts the level of robustness and $N$ represents the sample size.
Define the distribution set by the constraint on the likelihood ratio. With observed Data: $\xi_1, \xi_2, \ldots, \xi_N$, we define

$$D_N = \left\{ F_\xi \mid P(\xi \in \Xi) = 1, \quad L(\xi, F_\xi) \geq \gamma \right\}$$

where $\gamma$ adjusts the level of robustness and $N$ represents the sample size.

For example, assume the support of the uncertainty is finite

$$\xi_1, \xi_2, \ldots, \xi_n$$

and we observed $m_i$ samples on $\xi_i$. Then, $F_\xi$ has a finite discrete distribution $p_1, \ldots, p_n$ and

$$L(\xi, F_\xi) = \sum_{i=1}^{n} m_i \log p_i.$$. 
Theory on Likelihood Bounds

The model is a convex optimization problem, and connects to many statistical theories:

- Statistical Divergence theory: provide a bound on KL divergence
- Bayesian Statistics with the threshold \( \gamma \) estimated by samples: confidence level on the true distribution
- Non-parametric Empirical Likelihood theory: inference based on empirical likelihood by Owen
- Asymptotic Theory of the likelihood region
- Possible extensions to deal with Continuous Case

Wang, Glynn and Y [2012,2016]
DRO using Wasserstein Ambiguity Set

By the Kantorovich-Rubinstein theorem, the Wasserstein distance between two distributions can be expressed as the minimum cost of moving one to the other, which is a semi-infinite transportation LP.
By the Kantorovich-Rubinstein theorem, the Wasserstein distance between two distributions can be expressed as the minimum cost of moving one to the other, which is a semi-infinite transportation LP.

**Theorem**

*When using the Wasserstein ambiguity set*

\[ \mathcal{D}_N := \{ F_\xi \mid P(\xi \in \Xi) = 1 \& d(F_\xi, \hat{F}_N) \leq \varepsilon_N \}, \]

where \( d(F_1, F_2) \) is the Wasserstein distance function and \( N \) is the sample size, the DRO model satisfies the following properties:

- **Finite sample guarantee**: the correctness probability \( \bar{P}^N \) is high.
- **Asymptotic guarantee**: \( \bar{P}^\infty(\lim_{N \to \infty} \hat{x}_{\varepsilon_N} = x^*) = 1 \)
- **Tractability**: DRO is in the same complexity class as SAA

Mohajerin Esfahani & Kuhn [15, 17], Blanchet, Kang, Murthy [16], Duchi, Glynn, Namkoong
Let \( \{(\xi_i, \lambda_i)\}_{i=1}^{N} \) be a feature-label training set i.i.d. from \( P \), and consider applying logistic regression:

\[
\min_{x} \frac{1}{N} \sum_{i=1}^{N} \ell(x, \hat{\xi}_i, \hat{\lambda}_i) \text{ where } \ell(x, \xi, \lambda) = \ln(1 + \exp(-\lambda x^T \xi))
\]
DRO for Logistic Regression

Let \( \{(\hat{\xi}_i, \hat{\lambda}_i)\}_{i=1}^N \) be a feature-label training set i.i.d. from \( P \), and consider applying logistic regression:

\[
\min_x \frac{1}{N} \sum_{i=1}^N \ell(x, \hat{\xi}_i, \hat{\lambda}_i) \text{ where } \ell(x, \xi, \lambda) = \ln(1 + \exp(-\lambda x^T \xi))
\]

DRO suggests solving

\[
\min_x \sup_{F \in \mathcal{D}_N} \mathbb{E}_F[\ell(x, \xi_i, \lambda_i)]
\]

with the Wasserstein ambiguity set.
DRO for Logistic Regression

Let \( \{(\hat{\xi}_i, \hat{\lambda}_i)\}_{i=1}^{N} \) be a feature-label training set i.i.d. from \( P \), and consider applying logistic regression:

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} \ell(x, \hat{\xi}_i, \hat{\lambda}_i) \quad \text{where} \quad \ell(x, \xi, \lambda) = \ln(1 + \exp(-\lambda x^T \xi))
\]

DRO suggests solving

\[
\min_x \sup_{F \in \mathcal{D}_N} \mathbb{E}_F[\ell(x, \xi_i, \lambda_i)]
\]

with the Wasserstein ambiguity set.

When labels are considered to be error free, DRO with \( \mathcal{D}_N \) reduces to regularized logistic regression:

\[
\min_x \frac{1}{N} \sum_{i=1}^{N} \ell(x, \hat{\xi}_i, \hat{\lambda}_i) + \varepsilon \|x\|_*
\]
Results of the DRO Learning

Sinha, Namkoong and Duchi [2017]
Results of the DRO Learning: Original

<table>
<thead>
<tr>
<th>plane</th>
<th>car</th>
<th>bird</th>
<th>cat</th>
<th>deer</th>
<th>dog</th>
<th>frog</th>
<th>horse</th>
<th>ship</th>
<th>truck</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Plane" /></td>
<td><img src="image2.jpg" alt="Car" /></td>
<td><img src="image3.jpg" alt="Bird" /></td>
<td><img src="image4.jpg" alt="Cat" /></td>
<td><img src="image5.jpg" alt="Deer" /></td>
<td><img src="image6.jpg" alt="Dog" /></td>
<td><img src="image7.jpg" alt="Frog" /></td>
<td><img src="image8.jpg" alt="Horse" /></td>
<td><img src="image9.jpg" alt="Ship" /></td>
<td><img src="image10.jpg" alt="Truck" /></td>
</tr>
<tr>
<td><img src="image11.jpg" alt="Plane" /></td>
<td><img src="image12.jpg" alt="Car" /></td>
<td><img src="image13.jpg" alt="Bird" /></td>
<td><img src="image14.jpg" alt="Cat" /></td>
<td><img src="image15.jpg" alt="Deer" /></td>
<td><img src="image16.jpg" alt="Dog" /></td>
<td><img src="image17.jpg" alt="Frog" /></td>
<td><img src="image18.jpg" alt="Horse" /></td>
<td><img src="image19.jpg" alt="Ship" /></td>
<td><img src="image20.jpg" alt="Truck" /></td>
</tr>
<tr>
<td><img src="image21.jpg" alt="Plane" /></td>
<td><img src="image22.jpg" alt="Car" /></td>
<td><img src="image23.jpg" alt="Bird" /></td>
<td><img src="image24.jpg" alt="Cat" /></td>
<td><img src="image25.jpg" alt="Deer" /></td>
<td><img src="image26.jpg" alt="Dog" /></td>
<td><img src="image27.jpg" alt="Frog" /></td>
<td><img src="image28.jpg" alt="Horse" /></td>
<td><img src="image29.jpg" alt="Ship" /></td>
<td><img src="image30.jpg" alt="Truck" /></td>
</tr>
<tr>
<td><img src="image31.jpg" alt="Plane" /></td>
<td><img src="image32.jpg" alt="Car" /></td>
<td><img src="image33.jpg" alt="Bird" /></td>
<td><img src="image34.jpg" alt="Cat" /></td>
<td><img src="image35.jpg" alt="Deer" /></td>
<td><img src="image36.jpg" alt="Dog" /></td>
<td><img src="image37.jpg" alt="Frog" /></td>
<td><img src="image38.jpg" alt="Horse" /></td>
<td><img src="image39.jpg" alt="Ship" /></td>
<td><img src="image40.jpg" alt="Truck" /></td>
</tr>
</tbody>
</table>
Results of the DRO Learning: Nonrobust

<table>
<thead>
<tr>
<th>car</th>
<th>bird</th>
<th>cat</th>
<th>deer</th>
<th>dog</th>
<th>frog</th>
<th>horse</th>
<th>ship</th>
<th>truck</th>
<th>plane</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.jpg" alt="Image" /></td>
<td><img src="image2.jpg" alt="Image" /></td>
<td><img src="image3.jpg" alt="Image" /></td>
<td><img src="image4.jpg" alt="Image" /></td>
<td><img src="image5.jpg" alt="Image" /></td>
<td><img src="image6.jpg" alt="Image" /></td>
<td><img src="image7.jpg" alt="Image" /></td>
<td><img src="image8.jpg" alt="Image" /></td>
<td><img src="image9.jpg" alt="Image" /></td>
<td><img src="image10.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image11.jpg" alt="Image" /></td>
<td><img src="image12.jpg" alt="Image" /></td>
<td><img src="image13.jpg" alt="Image" /></td>
<td><img src="image14.jpg" alt="Image" /></td>
<td><img src="image15.jpg" alt="Image" /></td>
<td><img src="image16.jpg" alt="Image" /></td>
<td><img src="image17.jpg" alt="Image" /></td>
<td><img src="image18.jpg" alt="Image" /></td>
<td><img src="image19.jpg" alt="Image" /></td>
<td><img src="image20.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image21.jpg" alt="Image" /></td>
<td><img src="image22.jpg" alt="Image" /></td>
<td><img src="image23.jpg" alt="Image" /></td>
<td><img src="image24.jpg" alt="Image" /></td>
<td><img src="image25.jpg" alt="Image" /></td>
<td><img src="image26.jpg" alt="Image" /></td>
<td><img src="image27.jpg" alt="Image" /></td>
<td><img src="image28.jpg" alt="Image" /></td>
<td><img src="image29.jpg" alt="Image" /></td>
<td><img src="image30.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image31.jpg" alt="Image" /></td>
<td><img src="image32.jpg" alt="Image" /></td>
<td><img src="image33.jpg" alt="Image" /></td>
<td><img src="image34.jpg" alt="Image" /></td>
<td><img src="image35.jpg" alt="Image" /></td>
<td><img src="image36.jpg" alt="Image" /></td>
<td><img src="image37.jpg" alt="Image" /></td>
<td><img src="image38.jpg" alt="Image" /></td>
<td><img src="image39.jpg" alt="Image" /></td>
<td><img src="image40.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image41.jpg" alt="Image" /></td>
<td><img src="image42.jpg" alt="Image" /></td>
<td><img src="image43.jpg" alt="Image" /></td>
<td><img src="image44.jpg" alt="Image" /></td>
<td><img src="image45.jpg" alt="Image" /></td>
<td><img src="image46.jpg" alt="Image" /></td>
<td><img src="image47.jpg" alt="Image" /></td>
<td><img src="image48.jpg" alt="Image" /></td>
<td><img src="image49.jpg" alt="Image" /></td>
<td><img src="image50.jpg" alt="Image" /></td>
</tr>
<tr>
<td><img src="image51.jpg" alt="Image" /></td>
<td><img src="image52.jpg" alt="Image" /></td>
<td><img src="image53.jpg" alt="Image" /></td>
<td><img src="image54.jpg" alt="Image" /></td>
<td><img src="image55.jpg" alt="Image" /></td>
<td><img src="image56.jpg" alt="Image" /></td>
<td><img src="image57.jpg" alt="Image" /></td>
<td><img src="image58.jpg" alt="Image" /></td>
<td><img src="image59.jpg" alt="Image" /></td>
<td><img src="image60.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>
Results of the DRO Learning: DRO

<table>
<thead>
<tr>
<th>car</th>
<th>bird</th>
<th>cat</th>
<th>deer</th>
<th>dog</th>
<th>frog</th>
<th>horse</th>
<th>ship</th>
<th>truck</th>
<th>plane</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="car" /></td>
<td><img src="image" alt="bird" /></td>
<td><img src="image" alt="cat" /></td>
<td><img src="image" alt="deer" /></td>
<td><img src="image" alt="dog" /></td>
<td><img src="image" alt="frog" /></td>
<td><img src="image" alt="horse" /></td>
<td><img src="image" alt="ship" /></td>
<td><img src="image" alt="truck" /></td>
<td><img src="image" alt="plane" /></td>
</tr>
<tr>
<td><img src="image" alt="car" /></td>
<td><img src="image" alt="bird" /></td>
<td><img src="image" alt="cat" /></td>
<td><img src="image" alt="deer" /></td>
<td><img src="image" alt="dog" /></td>
<td><img src="image" alt="frog" /></td>
<td><img src="image" alt="horse" /></td>
<td><img src="image" alt="ship" /></td>
<td><img src="image" alt="truck" /></td>
<td><img src="image" alt="plane" /></td>
</tr>
<tr>
<td><img src="image" alt="car" /></td>
<td><img src="image" alt="bird" /></td>
<td><img src="image" alt="cat" /></td>
<td><img src="image" alt="deer" /></td>
<td><img src="image" alt="dog" /></td>
<td><img src="image" alt="frog" /></td>
<td><img src="image" alt="horse" /></td>
<td><img src="image" alt="ship" /></td>
<td><img src="image" alt="truck" /></td>
<td><img src="image" alt="plane" /></td>
</tr>
<tr>
<td><img src="image" alt="car" /></td>
<td><img src="image" alt="bird" /></td>
<td><img src="image" alt="cat" /></td>
<td><img src="image" alt="deer" /></td>
<td><img src="image" alt="dog" /></td>
<td><img src="image" alt="frog" /></td>
<td><img src="image" alt="horse" /></td>
<td><img src="image" alt="ship" /></td>
<td><img src="image" alt="truck" /></td>
<td><img src="image" alt="plane" /></td>
</tr>
</tbody>
</table>
Medical Application

Ref: Filtered Back Projection reconstructions of noise-free data
FBP: FBP reconstructions of noisy data
TV: TV-based reconstruction
DL: Dictionary Learning-based reconstruction
DL+DRO: DL+DRO to encourage low-rankness and robustness

Liu at all. [2017]
The DRO models yield a solution with a guaranteed confidence level to the possible distributions. Specifically, the confidence region of the distributions can be constructed upon the historical data and sample distributions.
Summary of DRO under Moment, Likelihood or Wasserstein Ambiguity Set

- The DRO models yield a solution with a guaranteed confidence level to the possible distributions. Specifically, the confidence region of the distributions can be constructed upon the historical data and sample distributions.

- The DRO models are tractable, and sometimes maintain the same computational complexity as the stochastic optimization models with known distribution.
Summary of DRO under Moment, Likelihood or Wasserstein Ambiguity Set

- The DRO models yield a solution with a guaranteed confidence level to the possible distributions. Specifically, the confidence region of the distributions can be constructed upon the historical data and sample distributions.
- The DRO models are tractable, and sometimes maintain the same computational complexity as the stochastic optimization models with known distribution.
- This approach can be applied to a wide range of problems, including inventory problems (e.g., newsvendor problem), portfolio selection problems, image reconstruction, machine learning, etc., with reported superior numerical results.
1. Introduction to Distributionally Robust Optimization

2. DRO under Moment, Likelihood and Wasserstein Bounds

3. Price of Correlation of High-Dimension Uncertainty

4. MDP Value-Iteration Sample/Computation Complexities

5. Online Linear Optimization and Dynamic Learning
Planning under High-Dimensional Stochastic Data

Portfolio Optimization

Facility Location
Planning under High-Dimensional Stochastic Data

Portfolio Optimization

\[ \text{minimize } \mathbb{E}_p[f(x, \xi)] \]

where \( \xi \) is a high-dimensional random vector, and many possible return/demand high-dimensional joint distributions.
Price of Correlations

One can also consider the distributionally robust approach:

\[
\min_{x \in X} \max_{p \in D} \mathbb{E}_p[f(x, \xi)]
\]

where \( D \) is the set of joint distributions such that the marginal distribution of \( \xi_i \) is \( p_i \) for each \( i \).

For simplicity, people are tempted to ignore correlations and assume independence among random variables (joint probability becomes the product of marginals). However, what is the risk associated with assuming independence? Can we analyze this risk in terms of properties of objective functions?

We precisely quantify this risk as \textit{Price of Correlations (POC)}

We provide tight bounds on POC for various cost functions.
Price of Correlations

One can also consider the distributionally robust approach:

\[
\min_{x \in X} \max_{p \in \mathcal{D}} \mathbb{E}_p[f(x, \xi)]
\]

where \( \mathcal{D} \) is the set of joint distributions such that the marginal distribution of \( \xi_i \) is \( p_i \) for each \( i \).

For simplicity, people are tempted to ignore correlations and assume independence among random variables (joint probability becomes the product of marginals). However, what is the risk associated with assuming independence? Can we analyze this risk in terms of properties of objective functions?
Price of Correlations

One can also consider the distributionally robust approach:

$$\min_{x \in X} \max_{p \in \mathcal{D}} \mathbb{E}_p[f(x, \xi)]$$

where $\mathcal{D}$ is the set of joint distributions such that the marginal distribution of $\xi_i$ is $p_i$ for each $i$.

For simplicity, people are tempted to ignore correlations and assume independence among random variables (joint probability becomes the product of marginals). However, what is the risk associated with assuming independence? Can we analyze this risk in terms of properties of objective functions?

- We precisely quantify this risk as **Price of Correlations (POC)**
- We provide tight bounds on POC for various cost functions.
Price of Correlations

Define

- \( \hat{x} \) be the optimal solution of stochastic program with independent distribution \( \hat{p}(\xi) = \prod_i p_i(\xi_i) \).

\[
\hat{x} = \arg \min_{x \in X} \mathbb{E}_{\hat{p}}[f(x, \xi)]
\]

- \( x^* \) be the optimal solution for the distributionally robust model.

\[
x^* = \arg \min_{x \in X} \max_{p \in \mathcal{D}} \mathbb{E}_p[f(x, \xi)]
\]

Then, Price of Correlations (POC), or Correlation Gap, is approximation ratio that \( \hat{x} \) achieves for distributionally robust model.

\[
POC = \frac{\max_{p \in \mathcal{D}} \mathbb{E}_p[f(\hat{x}, \xi)]}{\max_{p \in \mathcal{D}} \mathbb{E}_p[f(x^*, \xi)]}
\]
Price of Correlations

- Approximation of robust model
  - Minimax stochastic program can be replaced by stochastic program with independent distribution to get approximate solution.
  - Often easy to solve either by sampling or by other algorithmic techniques [e.g., Kleinberg et al. (1997), Möhring et al. (1999)]

Small POC means it is not too risky to assume independence.
Large POC suggests the importance of investing more on information gathering and learning the correlations in the joint distribution.
Approximation of robust model

- Minimax stochastic program can be replaced by stochastic program with independent distribution to get approximate solution.
- Often easy to solve either by sampling or by other algorithmic techniques [e.g., Kleinberg et al. (1997), Möhring et al. (1999)]

Captures “Value of Information”

- Small POC means it is not too risky to assume independence.
- Large POC suggests the importance of investing more on information gathering and learning the correlations in the joint distribution.
Price of Correlations

- Approximation of robust model
  - Minimax stochastic program can be replaced by stochastic program with independent distribution to get approximate solution.
  - Often easy to solve either by sampling or by other algorithmic techniques [e.g., Kleinberg et al. (1997), Möhring et al. (1999)]
- Captures “Value of Information”
  - Small POC means it is not too risky to assume independence.
  - Large POC suggests the importance of investing more on information gathering and learning the correlations in the joint distribution

Question: What function class has large POC? What function class has small POC?

- Submodularity leads to small POC
- Supermodularity leads to large POC
For any fixed $x$, function $f(\xi) = f(x, \xi)$ is submodular in random variable $\xi$

Decreasing marginal cost, economies of scale

$$f(\max\{\xi, \theta\}) + f(\min\{\xi, \theta\}) \leq f(\xi) + f(\theta)$$

For continuous functions: $\frac{\partial f(\xi)}{\partial \xi_i \partial \xi_j} \leq 0$
Submodularity Leads to Small POC

- For any fixed $x$, function $f(\xi) = f(x, \xi)$ is submodular in random variable $\xi$
- Decreasing marginal cost, economies of scale

$$f(\max\{\xi, \theta\}) + f(\min\{\xi, \theta\}) \leq f(\xi) + f(\theta)$$

- For continuous functions: $\frac{\partial f(\xi)}{\partial \xi_i \partial \xi_j} \leq 0$

**Theorem**

*If $f(\cdot, \xi)$ is monotone and submodular in $\xi$, then POC $\leq e/(e - 1)$.***

Calinescu, Chekuri, Pál, Vondrák [2007] for binary random variables, Agrawal, Ding, Saberi, Y, [2010] for general domains
Supermodularity Leads to Large POC

- For any fixed $x$, function $f(\xi) = f(x, \xi)$ is supermodular in random variable $\xi$
- Increasing marginal cost

$$\frac{\partial f(\xi)}{\partial \xi_i \partial \xi_j} \geq 0$$

e.g., effects of increase in congestion as demand increases.
- In worst case distribution large values of one variable will appear with large values of other variable – highly correlated
- We show example of supermodular set function with $\text{POC} = \Omega(2^n)$.

Agrawal, Ding, Saberi, Y, [2010]
Applications: Stochastic Bottleneck Matching

\[
\begin{align*}
\text{minimize}_{x \in X} & \quad \text{maximize}_{p \in P} \quad \mathbb{E}_p[\max_i \xi_i x_i] \\
\implies \quad & \text{minimize}_{x \in X} \quad \mathbb{E}_{\hat{p}}[\max_i \xi_i x_i]
\end{align*}
\]

where expected value is under independent distribution \( \hat{p} \).

- Monotone submodular function, \( e/(e - 1) \approx 1.6 \) approximation.
- Can be sampled efficiently, Chernoff type concentration bounds hold for monotone submodular functions.
- Reduces to a small convex problem

\[
\text{minimize}_{x \in X} \sum_{s \in S} \max_i \{s_i x_i\}
\]
Applications: Stochastic Bottleneck Matching

\[
\begin{align*}
\text{minimize}_{x \in X} & \quad \text{maximize}_{p \in P} \ E_p [\max_i \xi_i x_i] \\
\Rightarrow \\
\text{minimize}_{x \in X} & \quad E_{\hat{p}} [\max_i \xi_i x_i]
\end{align*}
\]

where expected value is under independent distribution \( \hat{p} \).

- Monotone submodular function, \( e/(e-1) \sim 1.6 \) approximation.
- Can be sampled efficiently, Chernoff type concentration bounds hold for monotone submodular functions.
- Reduces to a small convex problem

\[
\begin{align*}
\text{minimize}_{x \in X} & \quad \sum_{s \in S} \max_i \{s_i x_i\} \\
\Rightarrow \\
\text{minimize}_{x \in X} & \quad E_{\hat{p}} [||\xi^* x||_q]
\end{align*}
\]

where expected value is under independent distribution \( \hat{p} \).

- Monotone submodular function, \( e/(e-1) \sim 1.6 \) approximation.
Beyond Submodularity?

Monotone Subadditive Functions?
- Preserves economy of scale
- Example with
  \[ \text{POC} \geq \Omega\left(\sqrt{n} / \log \log(n)\right) \]

Fractionally Subadditive?
- \[ \text{POC} \geq \Omega\left(\sqrt{n} / \log \log(n)\right) \]
Beyond Submodularity?

Monotone Subadditive Functions?
- Preserves economy of scale
- Example with
  \[ \text{POC} = \Omega\left(\sqrt{n} / \log \log(n)\right) \]

Fractionally Subadditive?
- \[ \text{POC} \geq \Omega\left(\sqrt{n} / \log \log(n)\right) \]

Cost-sharing to the rescue
Cross-Monotone Cost-Sharing

A cooperative game theory concept

- Can cost $f(\xi_1, \ldots, \xi_n)$ be charged to participants $1, \ldots, n$ so that the share charged to participant $i$ decreases as the demands of other participants increase? [introduced by Thomson (1983, 1995) in context of bargaining]
- For submodular functions – charge marginal costs.
- $\beta$-approximate cost-sharing scheme: total cost charged is within $\beta$ of the original (expected) function value
Cross-Monotone Cost-Sharing

A cooperative game theory concept

- Can cost $f(\xi_1, \ldots, \xi_n)$ be charged to participants 1, \ldots, $n$ so that the share charged to participant $i$ decreases as the demands of other participants increase?

- For submodular functions – charge marginal costs.
- $\beta$-approximate cost-sharing scheme: total cost charged is within $\beta$ of the original (expected) function value

Approximate cost-sharing schemes exist for non-submodular functions

- 3-approximate cost-sharing for facility location cost function
  [Pál, Tardos 2003]

- 2-approximate cost-sharing for Steiner forest cost function
  [Könemann, Leonardi, Schäfer 2005]
Bounding POC via Cost-Sharing

**Theorem**

*If objective function $f(\cdot, \xi)$ is monotone in $\xi$ with $\beta$-cost-sharing scheme, $\text{POC} \leq 2\beta$.*

- $\text{POC} \leq 6$ for two-stage stochastic facility location problem
- $\text{POC} \leq 4$ for two-stage stochastic Steiner forest network design problem.

Agrawal, Ding, Saberi, Y, [2010]
The Cost-Sharing Condition is (near)-Tight

Theorem

If POC for function $f$ is less than $\beta$, there exists a cross-monotone cost-sharing scheme with expected $\beta$-budget balance.

We show examples of

- Monotone submodular function with POC $\geq \frac{e}{e-1}$.
- Facility location with POC $\geq 3$.
- Steiner tree network design with POC $\geq 2$.

Agrawal, Ding, Saberi, Y, [2010]
Summary of POC

- Characterizes the risk associated with assuming independence in a stochastic optimization problem.
- Can be upper bounded using properties of objective function.

Open questions
- Further characterizations of value of partial information in stochastic optimization problems
- Given partial information about correlations such as Covariance matrix
  - How does worst case distribution compare to maximum entropy distribution?
  - Block-wise independent distributions?
Summary of POC

- Characterizes the risk associated with assuming independence in a stochastic optimization problem.
- Can be upper bounded using properties of objective function.

Open questions

- Further characterizations of value of partial information in stochastic optimization problems
- Given partial information about correlations such as Covariance matrix
  - How does worst case distribution compare to maximum entropy distribution?
  - Block-wise independent distributions?
Table of Contents

1 Introduction to Distributionally Robust Optimization

2 DRO under Moment, Likelihood and Wasserstein Bounds

3 Price of Correlation of High-Dimension Uncertainty

4 MDP Value-Iteration Sample/Computation Complexities

5 Online Linear Optimization and Dynamic Learning
Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.
The Markov Decision/Game Process

- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.

- Markov game processes (MGPs) provide a mathematical slidework for modeling sequential decision-making of two-person turn-based zero-sum game.
Markov decision processes (MDPs) provide a mathematical framework for modeling **sequential** decision-making in situations where outcomes are partly **random** and partly under the control of a decision maker.

Markov game processes (MGPs) provide a mathematical slidework for modeling **sequential** decision-making of two-person turn-based zero-sum game.

MDGPs are useful for studying a wide range of optimization/game problems solved via **dynamic programming**, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).
The Markov Decision/Game Process

- Markov decision processes (MDPs) provide a mathematical framework for modeling sequential decision-making in situations where outcomes are partly random and partly under the control of a decision maker.

- Markov game processes (MGPs) provide a mathematical slidework for modeling sequential decision-making of two-person turn-based zero-sum game.

- MDGPs are useful for studying a wide range of optimization/game problems solved via dynamic programming, where it was known at least as early as the 1950s (cf. Shapley 1953, Bellman 1957).

- Modern applications include dynamic planning under uncertainty, reinforcement learning, social networking, and almost all other stochastic dynamic/sequential decision/game problems in Mathematical, Physical, Management and Social Sciences.
The Markov Decision/Game Process (continued)

- At each time step, the process is in some state \( i \in \{1, \ldots, m\} \)
  and the decision maker chooses an action \( j \in \mathcal{A}_i \) that is available in state \( i \).
The Markov Decision/Game Process (continued)

- At each time step, the process is in some state $i \in \{1, ..., m\}$ and the decision maker chooses an action $j \in \mathcal{A}_i$ that is available in state $i$.
- The process responds at the next time step by randomly moving into a new state $i'$, and giving the decision maker a corresponding cost or reward $c_j$. 
The Markov Decision/Game Process (continued)

- At each time step, the process is in some state \( i \in \{1, \ldots, m\} \) and the decision maker chooses an action \( j \in A_i \) that is available in state \( i \).
- The process responds at the next time step by randomly moving into a new state \( i' \), and giving the decision maker a corresponding cost or reward \( c_j \).
- The probability that the process changes from \( i \) to \( i' \) is influenced by the chosen action \( j \) in state \( i \). Specifically, it is given by the state transition probability distribution \( p_j \).
At each time step, the process is in some state $i \in \{1, \ldots, m\}$ and the decision maker chooses an action $j \in A_i$ that is available in state $i$.

The process responds at the next time step by randomly moving into a new state $i'$, and giving the decision maker a corresponding cost or reward $c_j$.

The probability that the process changes from $i$ to $i'$ is influenced by the chosen action $j$ in state $i$. Specifically, it is given by the state transition probability distribution $p_j$.

But given $i$ and $j$, the probability is conditionally independent of all previous states and actions. In other words, the state transitions of an MDP possess the Markov Property.
At each time step, the process is in some state \( i \in \{1, \ldots, m\} \) and the decision maker chooses an action \( j \in \mathcal{A}_i \) that is available in state \( i \).

The process responds at the next time step by randomly moving into a new state \( i' \), and giving the decision maker a corresponding cost or reward \( c_j \).

The probability that the process changes from \( i \) to \( i' \) is influenced by the chosen action \( j \) in state \( i \). Specifically, it is given by the state transition probability distribution \( p_j \).

But given \( i \) and \( j \), the probability is conditionally independent of all previous states and actions. In other words, the state transitions of an MDP possess the Markov Property.

A stationary policy for the decision maker is a function \( \pi = \{\pi_1, \pi_2, \cdots, \pi_m\} \) that specifies an action in each state, \( \pi_i \in \mathcal{A}_i \), that the decision maker will always choose; which also lead to a cost-to-go value for each state.
At each time step, the process is in some state \( i \in \{1, \ldots, m\} \) and the decision maker chooses an action \( j \in A_i \) that is available in state \( i \).

The process responds at the next time step by randomly moving into a new state \( i' \), and giving the decision maker a corresponding cost or reward \( c_j \).

The probability that the process changes from \( i \) to \( i' \) is influenced by the chosen action \( j \) in state \( i \). Specifically, it is given by the state transition probability distribution \( p_j \).

But given \( i \) and \( j \), the probability is conditionally independent of all previous states and actions. In other words, the state transitions of an MDP possess the Markov Property.

A stationary policy for the decision maker is a function \( \pi = \{\pi_1, \pi_2, \cdots, \pi_m\} \) that specifies an action in each state, \( \pi_i \in A_i \), that the decision maker will always choose; which also lead to a cost-to-go value for each state.
The Optimal Cost-to-Go Value Vector I

Let \( y \in \mathbb{R}^m \) represent the cost-to-go values of the \( m \) states, \( i \)th entry for \( i \)th state, of a given policy. The MDP problem entails choosing the optimal value vector \( y^* \) such that it satisfies:

\[
y_i^* = \min \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \quad \forall i,
\]

with optimal policy

\[
\pi_i^* = \arg \min \{ c_j + \gamma p_j^T y^*, \forall j \in A_i \}, \quad \forall i.
\]
The Optimal Cost-to-Go Value Vector I

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

$$y_i^* = \min\{c_j + \gamma p_j^T y^*, \forall j \in A_i\}, \forall i,$$

with optimal policy

$$\pi_i^* = \arg \min\{c_j + \gamma p_j^T y^*, \forall j \in A_i\}, \forall i.$$

In the Game setting, the conditions becomes:

$$y_i^* = \min\{c_j + \gamma p_j^T y^*, \forall j \in A_i\}, \forall i \in I^-,$$

and

$$y_i^* = \max\{c_j + \gamma p_j^T y^*, \forall j \in A_i\}, \forall i \in I^+.$$

They both are fix-point or saddle-point problems.
Let $y^0 \in \mathbb{R}^m$ represent the initial cost-to-go values of the $m$ states. The VI for MDP:

$$y_i^{k+1} = \min\{c_j + \gamma p_j^T y^k, \forall j \in A_i\}, \forall i.$$
Value-Iteration (VI) Method

Let $y^0 \in \mathbb{R}^m$ represent the initial cost-to-go values of the $m$ states. The VI for MDP:

$$y_i^{k+1} = \min\{c_j + \gamma p_j^Ty^k, \forall j \in A_i\}, \forall i.$$

The VI for MGP

$$y_i^{k+1} = \min\{c_j + \gamma p_j^Ty^k, \forall j \in A_i\}, \forall i \in I^-,$$

and

$$y_i^{k+1} = \max\{c_j + \gamma p_j^Ty^k, \forall j \in A_i\}, \forall i \in I^+.$$

The values inside the parenthesis are the so-called Q-values. Such operation can be written as

$$y^{k+1} = Ty^k.$$
Randomized/Sample-Based Value-Iteration

- In many practical applications, $p_j$ is unknown so that we have to approximate the mean $p_j^T y^k$ by stochastic sampling,
Randomized/Sample-Based Value-Iteration

- In many practical applications, $p_j$ is unknown so that we have to approximate the mean $p_j^T y^k$ by stochastic sampling.

- Even we know $p_j$ exactly, it may be too dense so that the computation of $p_j^T y^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.
Randomized/Sample-Based Value-Iteration

- In many practical applications, $p_j$ is unknown so that we have to approximate the mean $p_j^T y^k$ by stochastic sampling,
- Even we know $p_j$ exactly, it may be too dense so that the computation of $p_j^T y^k$ 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.
Randomized/Sample-Based Value-Iteration

- In many practical applications, \( p_j \) is unknown so that we have to approximate the mean \( p_j^T y^k \) by stochastic sampling.
- Even we know \( p_j \) exactly, it may be too dense so that the computation of \( p_j^T y^k \) 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.
- We analyze this performance using Hoeffdngs inequality and classic results on contraction properties of value iteration. Moreover, we improve the final result using Variance Reduction and Monotone Iteration.
In many practical applications, $p_j$ is unknown so that we have to approximate the mean $p_j^T y^k$ by stochastic sampling,

Even we know $p_j$ exactly, it may be too dense so that the computation of $p_j^T y^k$ 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.

We 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 Q-values so that the needed number of samples is decreased from iteration to iteration.
Two computation and sample complexity results are developed by (Sidford, Wang, Wu and Y [ICML 2017]) based on Variance Reduction (the VR technique has been used extensively in the design of fast stochastic methods for solving large-scale optimization problems in machine learning):

\[ \mathcal{O}\left( (mn + n(1 - \epsilon^2))^3 \log(\frac{1}{\epsilon}) \log(\frac{1}{\delta}) \right) \]

to compute an \( \epsilon \)-optimal policy with probability at least \( \frac{1}{\delta} \).

\[ \mathcal{O}\left( n(1 - \epsilon^2) \log(\frac{1}{\delta}) \right) \]

to compute an \( \epsilon \)-optimal policy with probability at least \( \frac{1}{\delta} \).

Sample complexity lower bound:

\[ \mathcal{O}\left( n(1 - \epsilon^2)^3 \right) \].
Randomized/Sample-Based Value-Iteration Results

Two computation and sample complexity results are developed by (Sidford, Wang, Wu and Y [ICML 2017]) based on Variance Reduction (the VR technique has been used extensively in the design of fast stochastic methods for solving large-scale optimization problems in machine learning):

- Knowing $p_j$:

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

to compute an $\epsilon$-optimal policy with probability at least $1 - \delta$. 
Randomized/Sample-Based Value-Iteration Results

Two computation and sample complexity results are developed by (Sidford, Wang, Wu and Y [ICML 2017]) based on **Variance Reduction** (the VR technique has been used extensively in the design of fast stochastic methods for solving large-scale optimization problems in machine learning):

- Knowing $p_j$:
  \[ O \left( (mn + \frac{n}{(1 - \gamma)^3}) \log\left(\frac{1}{\epsilon}\right) \log\left(\frac{1}{\delta}\right) \right) \]
  to compute an $\epsilon$-optimal policy with probability at least $1 - \delta$.

- Sampling on the pure generative model:
  \[ O \left( \frac{n}{(1 - \gamma)^4 \epsilon^2} \log\left(\frac{1}{\delta}\right) \right) \]
  to compute an $\epsilon$-optimal policy with probability at least $1 - \delta$. 
Randomized/Sample-Based Value-Iteration Results

Two computation and sample complexity results are developed by (Sidford, Wang, Wu and Y [ICML 2017]) based on Variance Reduction (the VR technique has been used extensively in the design of fast stochastic methods for solving large–scale optimization problems in machine learning):

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

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

- Sampling on the pure generative model:
  
  $$O \left( \frac{n}{(1 - \gamma)^4 \epsilon^2} \log\left(\frac{1}{\delta}\right) \right)$$

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

- Sample complexity lower bound: $O \left( \frac{n}{(1-\gamma)^3 \epsilon^2} \right)$. 
Summary of MDP Value-Iteration and Near-Optimal Randomized Value-Iteration Result

[Sidford, Wang, Wu, Yang and Ye NIPS 2018].

- 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 \( \epsilon \)-optimal policy with probability at least \( 1 - \delta \).
[Sidford, Wang, Wu, Yang and Y NIPS 2018].

- Computation and sample complexity on the pure generative model:

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

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

- The method is also extended to computing $\epsilon$-optimal policies for finite-horizon MDP with a generative model and provide a nearly matching sample complexity lower bound.
Summary of MDP Value-Iteration and Near-Optimal Randomized Value-Iteration Result

[Sidford, Wang, Wu, Yang and Y NIPS 2018].

- 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 \( \epsilon \)-optimal policy with probability at least \( 1 - \delta \).

- The method is also extended to computing \( \epsilon \)-optimal policies for finite-horizon MDP with a generative model and provide a nearly matching sample complexity lower bound.

- More recently, the result is substantially generalized to the Stochastic Game process.
Table of Contents

1. Introduction to Distributionally Robust Optimization
2. DRO under Moment, Likelihood and Wasserstein Bounds
3. Price of Correlation of High-Dimension Uncertainty
4. MDP Value-Iteration Sample/Computation Complexities
5. Online Linear Optimization and Dynamic Learning
Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
Background

Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices
Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices
- Decision: To sell or not to each individual customer?
Consider a store that sells a number of goods/products

- There is a fixed selling period or number of buyers
- There is a fixed inventory of goods
- Customers come and require a bundle of goods and bid for certain prices
- Decision: To sell or not to each individual customer?
- Objective: Maximize the revenue.
An Example

<table>
<thead>
<tr>
<th></th>
<th>Bid 1 ($t = 1$)</th>
<th>Bid 2 ($t = 2$)</th>
<th>.....</th>
<th>Inventory ($b$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price ($\pi_t$)</strong></td>
<td>$100$</td>
<td>$30$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Decision</strong></td>
<td>$x_1$</td>
<td>$x_2$</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>Pants</strong></td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>100</td>
</tr>
<tr>
<td><strong>Shoes</strong></td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>50</td>
</tr>
<tr>
<td><strong>T-shirts</strong></td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>500</td>
</tr>
<tr>
<td><strong>Jackets</strong></td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>200</td>
</tr>
<tr>
<td><strong>Hats</strong></td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1000</td>
</tr>
</tbody>
</table>
The classical offline version of the above program can be formulated as a linear (integer) program as all information data would have arrived: compute $x_t$, $t = 1, \ldots, n$ and

\[
\text{maximize}_x \quad \sum_{t=1}^{n} \pi_t x_t \\
\text{subject to} \quad \sum_{t=1}^{n} a_{it} x_t \leq b_i, \quad \forall i = 1, \ldots, m \\
x_t \in \{0, 1\} \quad (0 \leq x_t \leq 1), \quad \forall t = 1, \ldots, n.
\]
Online Linear Programming Model

The classical **offline** version of the above program can be formulated as a linear (integer) program as all information data would have arrived: compute $x_t$, $t = 1, \ldots, n$ and

\[
\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{n} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{n} a_{it} x_t \leq b_i, \quad \forall i = 1, \ldots, m \\
& \quad x_t \in \{0, 1\} \ (0 \leq x_t \leq 1), \quad \forall t = 1, \ldots, n.
\end{align*}
\]

Now we consider the **online or streamline** and **data-driven** version of this problem:

- We only know $b$ and $n$ at the start
Online Linear Programming Model

The classical offline version of the above program can be formulated as a linear (integer) program as all information data would have arrived: compute $x_t$, $t = 1, \ldots, n$ and

$$\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{n} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{n} a_{it} x_t \leq b_i, \quad \forall i = 1, \ldots, m \\
& \quad x_t \in \{0, 1\} \ (0 \leq x_t \leq 1), \quad \forall t = 1, \ldots, n.
\end{align*}$$

Now we consider the online or streamline and data-driven version of this problem:

- We only know $b$ and $n$ at the start
- the bidder information is revealed sequentially along with the corresponding objective coefficient.
Online Linear Programming Model

The classical offline version of the above program can be formulated as a linear (integer) program as all information data would have arrived: compute $x_t$, $t = 1, ..., n$ and

$$\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{n} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{n} a_{it} x_t \leq b_i, \quad \forall i = 1, \ldots, m \\
& \quad x_t \in \{0, 1\} \ (0 \leq x_t \leq 1), \quad \forall t = 1, \ldots, n.
\end{align*}$$

Now we consider the online or streamline and data-driven version of this problem:

- We only know $b$ and $n$ at the start
- the bidder information is revealed sequentially along with the corresponding objective coefficient.
- an irrevocable decision must be made as soon as an order arrives without observing or knowing the future data.
Model Assumptions

Main Assumptions

- $0 \leq a_{it} \leq 1$, for all $(i, t)$;
- $\pi_t \geq 0$ for all $t$
- The bids $(a_t, \pi_t)$ arrive in a random order (rather than from some iid distribution).
Model Assumptions

Main Assumptions

- \( 0 \leq a_{it} \leq 1 \), for all \((i, t)\);
- \( \pi_t \geq 0 \) for all \(t\)
- The bids \((a_t, \pi_t)\) arrive in a random order (rather than from some iid distribution).

Denote the offline LP maximal value by \( \text{OPT}(A, \pi) \). We call an online algorithm \(A\) to be \(c\)-competitive if and only if

\[
E_\sigma \left[ \sum_{t=1}^{n} \pi_t x_t(\sigma, A) \right] \geq c \cdot \text{OPT}(A, \pi) \quad \forall (A, \pi),
\]

where \(\sigma\) is the permutation of arriving orders.

In what follows, we let

\[
B = \min_i b_i (> 0).
\]
Main Results: Necessary and Sufficient Conditions

**Theorem**

For any fixed $0 < \epsilon < 1$, there is no online algorithm for solving the linear program with competitive ratio $1 - \epsilon$ if

$$B < \frac{\log(m)}{\epsilon^2}.$$
Main Results: Necessary and Sufficient Conditions

Theorem
For any fixed $0 < \epsilon < 1$, there is no online algorithm for solving the linear program with competitive ratio $1 - \epsilon$ if

$$B < \frac{\log(m)}{\epsilon^2}.$$ 

Theorem
For any fixed $0 < \epsilon < 1$, there is a $1 - \epsilon$ competitive online algorithm for solving the linear program if

$$B \geq \Omega\left(\frac{m \log(n/\epsilon)}{\epsilon^2}\right).$$
Theorem
For any fixed $0 < \epsilon < 1$, there is no online algorithm for solving the linear program with competitive ratio $1 - \epsilon$ if

$$B < \frac{\log(m)}{\epsilon^2}.$$ 

Theorem
For any fixed $0 < \epsilon < 1$, there is a $1 - \epsilon$ competitive online algorithm for solving the linear program if

$$B \geq \Omega\left(\frac{m \log (n/\epsilon)}{\epsilon^2}\right).$$

Consider $m = 1$ and inventory level $B$, one can construct an instance where $OPT = B$, and there will be a loss of $\sqrt{B}$ with a high probability, which give an approximation ratio $1 - \frac{1}{\sqrt{B}}$. 
Ideas to Prove Negative Result

- Consider $m = 1$ and inventory level $B$, one can construct an instance where $OPT = B$, and there will be a loss of $\sqrt{B}$ with a high probability, which give an approximation ratio $1 - \frac{1}{\sqrt{B}}$.

- Consider general $m$ and inventory level $B$ for each good. We are able to construct an instance to decompose the problem into $\log(m)$ separable problems, each of which has an inventory level $B/\log(m)$ on a composite “single good” and $OPT = B/\log(m)$. 
Consider $m = 1$ and inventory level $B$, one can construct an instance where $OPT = B$, and there will be a loss of $\sqrt{B}$ with a high probability, which give an approximation ratio $1 - \frac{1}{\sqrt{B}}$.

Consider general $m$ and inventory level $B$ for each good. We are able to construct an instance to decompose the problem into $\log(m)$ separable problems, each of which has an inventory level $B/\log(m)$ on a composite “single good” and $OPT = B/\log(m)$.

Then, with high probability each “single good” case has a loss of $\sqrt{B/\log(m)}$ and the total loss of $\sqrt{B \cdot \log(m)}$. Thus, approximation ratio is at best $1 - \frac{\sqrt{\log(m)}}{\sqrt{B}}$. 
Necessary Result I

Multidimensional knapsack

\[ B \]

\[ m \] dimensional

\[ \log(m) \] type of MUST-HAVE items
(profit = 4)

\[ 3 \log(m) \] type of NORMAL items
(profit = 3, 2, 1)

\[ B/\log(m) \] numbers of each

Ye, Yinyu (Stanford) Robust and Online Optimization November 14, 2018 52 / 66
Multidimensional knapsack

Once a MUST-HAVE item is picked, only the corresponding NORMAL items can fit in that column.

$log(m)$ type of MUST-HAVE items (profit = 4)

$3 \log(m)$ type of NORMAL items (profit = 3, 2, 1)

$B/\log(m)$ numbers of each
The proof of the positive result is *constructive* and based on a learning policy.
Ideas to Prove Positive Result: Dynamic Learning

The proof of the positive result is **constructive** and based on a learning policy.

- There is no distribution known so that any type of **stochastic optimization** models is not applicable.
Ideas to Prove Positive Result: Dynamic Learning

The proof of the positive result is constructive and based on a learning policy.

- There is no distribution known so that any type of stochastic optimization models is not applicable.
- Unlike dynamic programming, the decision maker does not have full information/data so that a backward recursion can not be carried out to find an optimal sequential decision policy.
Ideas to Prove Positive Result: Dynamic Learning

The proof of the positive result is **constructive** and based on a learning policy.

- There is no distribution known so that any type of **stochastic optimization** models is not applicable.
- Unlike dynamic programming, the decision maker does not have full information/data so that a **backward recursion** cannot be carried out to find an optimal sequential decision policy.
- Thus, the online algorithm needs to be **learning-based**, in particular, **learning-while-doing**.
The problem would be easy if there are "ideal prices":

<table>
<thead>
<tr>
<th>Bid 1((t = 1))</th>
<th>Bid 2((t = 2))</th>
<th>.....</th>
<th>Inventory((b))</th>
<th>(p^*)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bid((\pi_t))</td>
<td>$100</td>
<td>$30</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Decision</td>
<td>(x_1)</td>
<td>(x_2)</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Pants</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>100</td>
</tr>
<tr>
<td>Shoes</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>50</td>
</tr>
<tr>
<td>T-shirts</td>
<td>0</td>
<td>1</td>
<td>...</td>
<td>500</td>
</tr>
<tr>
<td>Jackets</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>200</td>
</tr>
<tr>
<td>Hats</td>
<td>1</td>
<td>1</td>
<td>...</td>
<td>1000</td>
</tr>
</tbody>
</table>
Pricing the bid: The optimal dual price vector $p^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $\pi_t > a_t^T p^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.
Pricing the bid: The optimal dual price vector $p^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $\pi_t > a_t^T p^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.

Based on this observation, our online algorithm works by learning a threshold price vector $\hat{p}$ and using $\hat{p}$ to price the bids.
Pricing the bid: The optimal dual price vector $p^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $\pi_t > a_t^T p^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.

Based on this observation, our online algorithm works by learning a threshold price vector $\hat{p}$ and using $\hat{p}$ to price the bids.

One-time learning algorithm: learn the price vector once using the initial $\epsilon n$ input.
Pricing the bid: The optimal dual price vector \( p^* \) of the offline LP problem can play such a role, that is \( x_t^* = 1 \) if \( \pi_t > a_t^T p^* \) and \( x_t^* = 0 \) otherwise, yields a near-optimal solution.

Based on this observation, our online algorithm works by learning a threshold price vector \( \hat{p} \) and using \( \hat{p} \) to price the bids.

One-time learning algorithm: learn the price vector once using the initial \( \epsilon n \) input.

Dynamic learning algorithm: dynamically update the prices at a carefully chosen pace.
We illustrate a simple One-Time Learning Algorithm:

- Set $x_t = 0$ for all $1 \leq t \leq \epsilon n$;
One-Time Learning Algorithm

We illustrate a simple One-Time Learning Algorithm:

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

\[
\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{\epsilon n} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{\epsilon n} a_{it} x_t \leq (1 - \epsilon) \epsilon b_i \quad i = 1, \ldots, m \\
& \quad 0 \leq x_t \leq 1 \quad t = 1, \ldots, \epsilon n
\end{align*}
\]

and get the optimal dual solution $\hat{p}$;
One-Time Learning Algorithm

We illustrate a simple One-Time Learning Algorithm:

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

\[
\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{\epsilon n} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{\epsilon n} a_{it} x_t \leq (1 - \epsilon)\epsilon b_i \quad i = 1, \ldots, m \\
& \quad 0 \leq x_t \leq 1 \quad t = 1, \ldots, \epsilon n
\end{align*}
\]

and get the optimal 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 $x_t = 0$. 

Ye, Yinyu (Stanford)
Robust and Online Optimization
November 14, 2018 57 / 66
One-Time Learning Algorithm Result

Theorem

For any fixed $\epsilon > 0$, the one-time learning algorithm is $(1 - \epsilon)$ competitive for solving the linear program when

$$B \geq \Omega \left( \frac{m \log (n/\epsilon)}{\epsilon^3} \right)$$

This is one $\epsilon$ worse than the optimal bound.
Outline of the Proof

- With high probability, we clear the market;
- With high probability, the revenue is near-optimal if we include the initial $\epsilon$ portion revenue;
- With high probability, the first $\epsilon$ portion revenue, a learning cost, doesn’t contribute too much.

Then, we prove that the one-time learning algorithm is $(1 - \epsilon)$ competitive under condition $B \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$.
Outline of the Proof

- With high probability, we clear the market;
- With high probability, the revenue is near-optimal if we include the initial $\epsilon$ portion revenue;
- With high probability, the first $\epsilon$ portion revenue, a learning cost, doesn’t contribute too much.

Then, we prove that the one-time learning algorithm is $(1 - \epsilon)$ competitive under condition $B \geq \frac{6m \log(n/\epsilon)}{\epsilon^3}$.

Again, this is one $\epsilon$ factor worse than the lower bound...
In the dynamic price learning algorithm, we update the price at time $\epsilon n$, $2\epsilon n$, $4\epsilon n$, ..., till $2^k \epsilon \geq 1$. 
Dynamic Learning Algorithm

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

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

\[
\begin{align*}
\text{maximize}_x & \quad \sum_{t=1}^{\ell} \pi_t x_t \\
\text{subject to} & \quad \sum_{t=1}^{\ell} a_{it} x_t \leq (1 - h_\ell) \frac{\ell}{n} b_i & i = 1, \ldots, m \\
& \quad 0 \leq x_t \leq 1 & t = 1, \ldots, \ell
\end{align*}
\]

where

\[
h_\ell = \epsilon \sqrt{\frac{n}{\ell}};
\]

and this price vector is used to determine the allocation for the next immediate period.
Geometric Pace/Grid of Price Updating

$t_1 = \varepsilon n$

$t_2 = 2\varepsilon n$

$t_3 = 4\varepsilon n$

$t_4 = 8\varepsilon n$
In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon.
In the dynamic algorithm, we update the prices $\log_2 (1/\epsilon)$ times during the entire time horizon.

The numbers $h_\ell$ play an important role in improving the condition on $B$ in the main theorem. It basically balances the probability that the inventory ever gets violated and the lost of revenue due to the factor $1 - h_\ell$. 
Comments on Dynamic Learning Algorithm

In the dynamic algorithm, we update the prices $\log_2 \left( \frac{1}{\epsilon} \right)$ times during the entire time horizon.

The numbers $h_\ell$ play an important role in improving the condition on $B$ in the main theorem. It basically balances the probability that the inventory ever gets violated and the lost of revenue due to the factor $1 - h_\ell$.

Choosing large $h_\ell$ (more conservative) at the beginning periods and smaller $h_\ell$ (more aggressive) at the later periods, one can now control the loss of revenue by an $\epsilon$ order while the required size of $B$ can be weakened by an $\epsilon$ factor.
## Related Work on Random-Permutation

<table>
<thead>
<tr>
<th>Author/Year</th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>$B \geq \frac{1}{e^2}$, for $m = 1$</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>
## Related Work on Random-Permutation

<table>
<thead>
<tr>
<th>Reference</th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>$B \geq \frac{1}{\epsilon^2}$, for $m = 1$</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Devanur et al [2009]</td>
<td>$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$</td>
<td>One-time</td>
</tr>
</tbody>
</table>

Table: Comparison of several existing results.
Related Work on Random-Permutation

<table>
<thead>
<tr>
<th>Author(s)</th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>$B \geq \frac{1}{\epsilon^2}$, for $m = 1$</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Devanur et al [2009]</td>
<td>$\text{OPT} \geq \frac{m^2 \log(n)}{\epsilon^3}$</td>
<td>One-time</td>
</tr>
<tr>
<td>Feldman et al [2010]</td>
<td>$B \geq \frac{m \log n}{\epsilon^3}$ and $\text{OPT} \geq \frac{m \log n}{\epsilon}$</td>
<td>One-time</td>
</tr>
</tbody>
</table>
## Related Work on Random-Permutation

<table>
<thead>
<tr>
<th></th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>$B \geq \frac{1}{\epsilon^2}$, for $m = 1$</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Devanur et al [2009]</td>
<td></td>
<td>One-time</td>
</tr>
<tr>
<td>Feldman et al [2010]</td>
<td>$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$</td>
<td>One-time</td>
</tr>
<tr>
<td>Agrawal et al [2010]</td>
<td>$B \geq \frac{m \log n}{\epsilon^3}$ and $OPT \geq \frac{m \log n}{\epsilon}$</td>
<td>Dynamic</td>
</tr>
<tr>
<td></td>
<td>$B \geq \frac{m \log n}{\epsilon^2}$ or $OPT \geq \frac{m^2 \log n}{\epsilon^2}$</td>
<td></td>
</tr>
</tbody>
</table>
## Related Work on Random-Permutation Sufficient Condition

<table>
<thead>
<tr>
<th>Source</th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>$B \geq \frac{1}{\epsilon^2}$, for $m = 1$</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Devanur et al [2009]</td>
<td>$\text{OPT} \geq \frac{m^2 \log(n)}{\epsilon^3}$</td>
<td>One-time</td>
</tr>
<tr>
<td>Feldman et al [2010]</td>
<td>$B \geq \frac{m \log n}{\epsilon^3}$ and $\text{OPT} \geq \frac{m \log n}{\epsilon}$</td>
<td>One-time</td>
</tr>
<tr>
<td>Agrawal et al [2010]</td>
<td>$B \geq \frac{m \log n}{\epsilon^2}$ or $\text{OPT} \geq \frac{m^2 \log n}{\epsilon^2}$</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Molinaro/Ravi [2013]</td>
<td>$B \geq \frac{m^2 \log m}{\epsilon^2}$</td>
<td>Dynamic</td>
</tr>
</tbody>
</table>
### Related Work on Random-Permutation Learning

<table>
<thead>
<tr>
<th>Study</th>
<th>Sufficient Condition</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kleinberg [2005]</td>
<td>( B \geq \frac{1}{\epsilon^2}, \text{ for } m = 1 )</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Devanur et al [2009]</td>
<td>( OPT \geq \frac{m^2 \log(n)}{\epsilon^3} )</td>
<td>One-time</td>
</tr>
<tr>
<td>Feldman et al [2010]</td>
<td>( B \geq \frac{m \log n}{\epsilon^3} ) and ( OPT \geq \frac{m \log n}{\epsilon} )</td>
<td>One-time</td>
</tr>
<tr>
<td>Agrawal et al [2010]</td>
<td>( B \geq \frac{m \log n}{\epsilon^2} ) or ( OPT \geq \frac{m^2 \log n}{\epsilon^2} )</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Molinaro/Ravi [2013]</td>
<td>( B \geq \frac{m \log m}{\epsilon^2} )</td>
<td>Dynamic</td>
</tr>
<tr>
<td>Kesselheim et al [2014]</td>
<td>( B \geq \frac{\log m}{\epsilon^2} )</td>
<td>Dynamic*</td>
</tr>
<tr>
<td>Gupta/Molinaro [2014]</td>
<td>( B \geq \frac{\log m}{\epsilon^2} )</td>
<td>Dynamic*</td>
</tr>
<tr>
<td>Agrawal/Devanur [2014]</td>
<td>( B \geq \frac{\log m}{\epsilon^2} )</td>
<td>Dynamic*</td>
</tr>
</tbody>
</table>

**Table:** Comparison of several existing results
Selling a good in a fixed horizon $T$, and there is no salvage value for the remaining quantities after the horizon.
Selling a good in a fixed horizon $T$, and there is no salvage value for the remaining quantities after the horizon.

The production lead time is long so that the inventory $B$ is fixed and can not be replenished during the selling season.
Selling a good in a fixed horizon $T$, and there is no salvage value for the remaining quantities after the horizon. The production lead time is long so that the inventory $B$ is fixed and cannot be replenished during the selling season. Demand arrives in a Poisson process, where the arrival rate $\lambda(p)$ depends only on the instantaneous price posted by the seller.
Selling a good in a fixed horizon $T$, and there is no salvage value for the remaining quantities after the horizon.

The production lead time is long so that the inventory $B$ is fixed and can not be replenished during the selling season.

Demand arrives in a Poisson process, where the arrival rate $\lambda(p)$ depends only on the instantaneous price posted by the seller.

Objective is to maximize the expected revenue.

Near optimal algorithm found for the one good case (Wang, Deng and Y [2014]).
Geometric Pace of Price Testing

Iter 1
\( \tau_1 = 0.0035; \)
\( \kappa_1 = 12; \)
\( p \) range: 0.1-10
\( p^u = 5.05 \)
\( p^c = 3.4 \)

Iter 3
\( \tau_3 = 0.15 \)
\( \kappa_3 = 5; \)
\( p \) range: 4.90-5.35
\( p^u = 5.07 \)
\( p^c = 4.90 \)

Iter 2
\( \tau_2 = 0.037; \)
\( \kappa_2 = 7; \)
\( p \) range: 3.99-5.69
\( p^u = 5.21 \)
\( p^c = 3.99 \)

Iter 4
\( \tau_4 = 0.35; \)
\( \kappa_4 = 4; \)
\( p \) range: 4.96-5.18
\( p^u = 5.01 \)
\( p^c = 4.96 \)

\( \tau \): Time spent in each period
\( \kappa \): Number of price tested

Apply price 5.01 until the end

\( T \)
\[ B = \frac{\log m}{\epsilon^2} \] is now a necessary and sufficient condition (differing by a constant factor).
\[ B = \frac{\log m}{\epsilon^2} \] is now a necessary and sufficient condition (differing by a constant factor).

Thus, they are optimal online algorithms for a very general class of online linear programs.
Summary and Future Questions on Online Optimization

- $B = \frac{\log m}{\epsilon^2}$ is now a **necessary and sufficient** condition (differing by a **constant** factor).
- Thus, they are **optimal** online algorithms for a very general class of online linear programs.
- The algorithms are **distribution-free and/or non-parametric**, thereby robust to distribution/data uncertainty.
Summary and Future Questions on Online Optimization

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are optimal online algorithms for a very general class of online linear programs.
- The algorithms are distribution-free and/or non-parametric, thereby robust to distribution/data uncertainty.
- The dynamic learning has the feature of "learning-while-doing", and is provably better than one-time learning by one factor.
Summary and Future Questions on Online Optimization

- \( B = \frac{\log m}{\epsilon^2} \) is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are optimal online algorithms for a very general class of online linear programs.
- The algorithms are distribution-free and/or non-parametric, thereby robust to distribution/data uncertainty.
- The dynamic learning has the feature of “learning-while-doing”, and is provably better than one-time learning by one factor.
- Buy-and-sell or double market?
Summary and Future Questions on Online Optimization

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are optimal online algorithms for a very general class of online linear programs.
- The algorithms are distribution-free and/or non-parametric, thereby robust to distribution/data uncertainty.
- The dynamic learning has the feature of “learning-while-doing”, and is provably better than one-time learning by one factor.
- Buy-and-sell or double market?
- Multi-item price-posting market?
Summary and Future Questions on Online Optimization

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are optimal online algorithms for a very general class of online linear programs.
- The algorithms are distribution-free and/or non-parametric, thereby robust to distribution/data uncertainty.
- The dynamic learning has the feature of “learning-while-doing”, and is provably better than one-time learning by one factor.
- Buy-and-sell or double market?
- Multi-item price-posting market?
- More general online optimization?