Distributionally Robust Stochastic and Online Optimization Driven by Data/Samples

Models/Algorithms for Learning and Decision Making

Yinyu Ye

[1]Department 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

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.

Table of Contents

Introduction to DRO

We start from considering a stochastic optimization problem as follows:

$$\text{maximize}_{\mathbf{x} \in X} \quad \mathbb{E}_{F_\xi}[h(\mathbf{x}, \xi)] \tag{1}$$

where $\mathbf{x}$ is the decision variable with feasible region X, ξ represents random variables satisfying joint distribution F_ξ.

Introduction to DRO

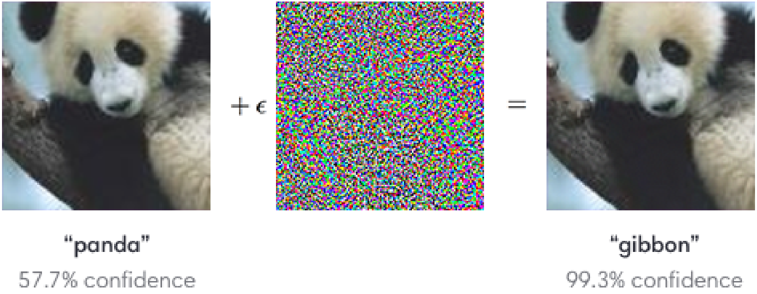
We start from considering a stochastic optimization problem as follows:

$$\text{maximize}_{\mathbf{x}\in X} \quad \mathbb{E}_{F_\xi}[h(\mathbf{x},\xi)] \tag{1}$$

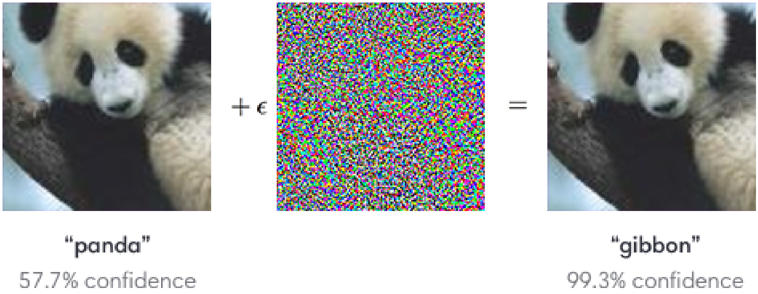
where $\mathbf{x}$ is the decision variable with feasible region X, ξ represents random variables satisfying joint distribution F_ξ.

- Pros: In many cases, the expected value is a good measure of performance
- Cons: One has to know the exact distribution of ξ 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



Learning with Noises



Goodfellow et al. [2014]

Robust Optimization

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

$$\text{maximize}_{\mathbf{x} \in X} \quad \min_{\xi \in \Xi} h(\mathbf{x}, \xi) \tag{2}$$

where Ξ is the support of ξ.

Robust Optimization

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

$$\text{maximize}_{\mathbf{x}\in X} \quad \min_{\xi\in\Xi} h(\mathbf{x},\xi) \tag{2}$$

where Ξ is the support of ξ.

- 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.

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.

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:

$$\text{maximize}_{\mathbf{x}\in X} \quad \min_{F_\xi \in \mathcal{D}} \mathbb{E}_{F_\xi}[h(\mathbf{x},\xi)] \tag{3}$$

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

Distributionally Robust Optimization

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

$$\text{maximize}_{\mathbf{x}\in X} \quad \min_{F_\xi \in \mathcal{D}} \mathbb{E}_{F_\xi}[h(\mathbf{x},\xi)] \tag{3}$$

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

When choosing $\mathcal{D}$, 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]
- Lagoa and Barmish [2002] and Shapiro [2006] simply considers a set containing unimodal distributions, Kleinberg et al. [1997] and M'ohring et al. [1999] considers the product distribution

Table of Contents

DRO with Moment Bounds

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

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

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 ϵ in time polynomial in $\log(1/\epsilon)$ and the sizes of $\mathbf{x}$ and ξ

Delage and Y [2010]

Confidence Region on F_ξ

Does the construction of $\mathcal{D}$ make a statistical sense?

Confidence Region on F_ξ

Does the construction of $\mathcal{D}$ make a statistical sense?

Theorem

Consider

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

where μ_0 and Σ_0 are point estimates from the empirical data (of size m) and Ξ lies in a ball of radius R such that $||\xi||_2 \leq R$ a.s..

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

$$P(F_\xi \in D(\gamma_1, \gamma_2)) \geq 1 - \delta$$

DRO with Likelihood Bounds

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

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

where γ adjusts the level of robustness and N represents the sample size.

DRO with Likelihood Bounds

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

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

where γ adjusts the level of robustness and N represents the sample size.

For example, assume the support of the uncertainty is finite

$$\xi_1, \xi_2, ...\xi_n$$

and we observed m_i samples on ξ_i. Then, F_ξ has a finite discrete distribution $p_1, ..., p_n$ 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 γ 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.

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.

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

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 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 :

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

- DRO suggests solving

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

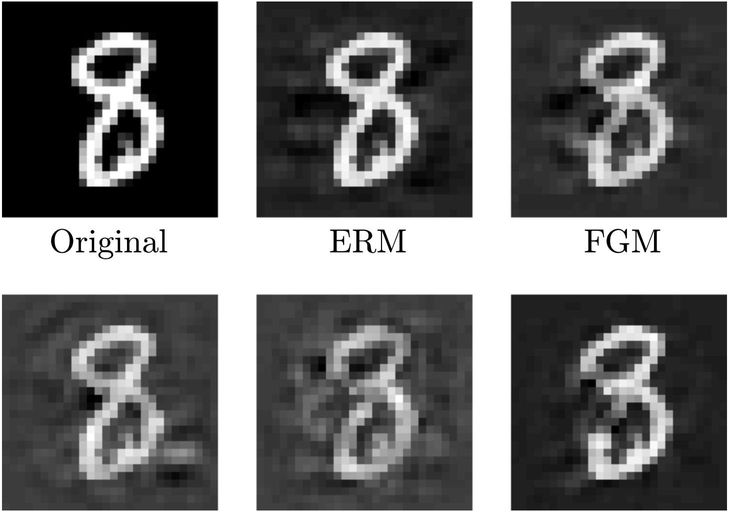
with the Wasserstein ambiguity set.

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

$$\min_x \frac{1}{N}\sum_{i=1}^N \ell(x, \hat{\xi}_i, \hat{\lambda}_i) + \varepsilon\|x\|_*$$

Shafieezadeh Abadeh, Mohajerin Esfahani, & Kuhn, NIPS, [2015]

Results of the DRO Learning

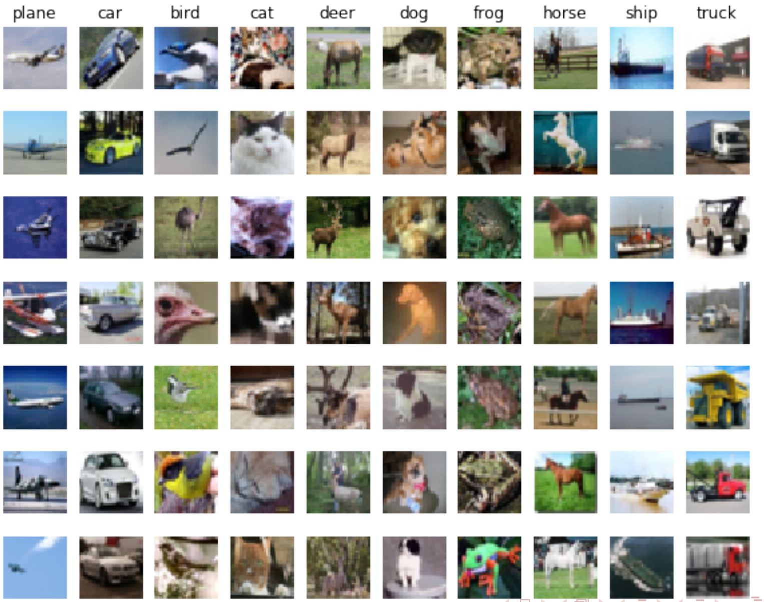


Original ERM FGM

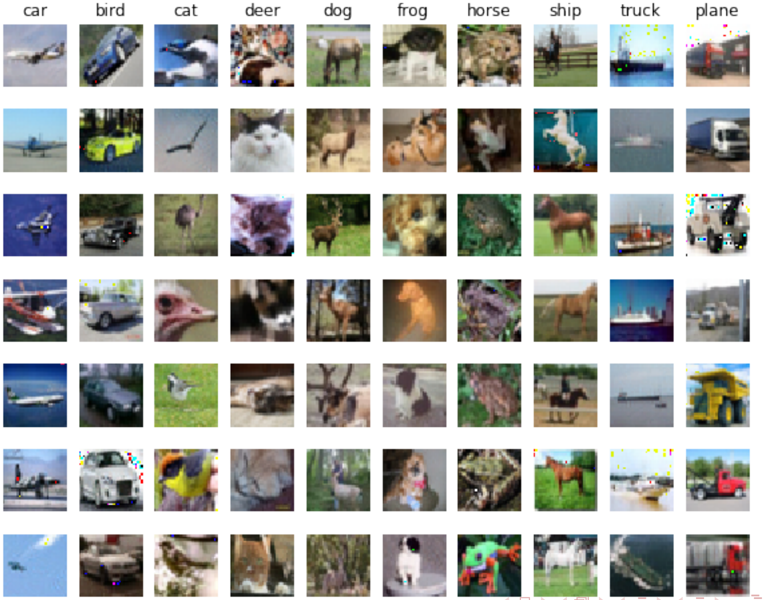
IFGM PGM WRM

Sinha, Namkoong and Duchi [2017]

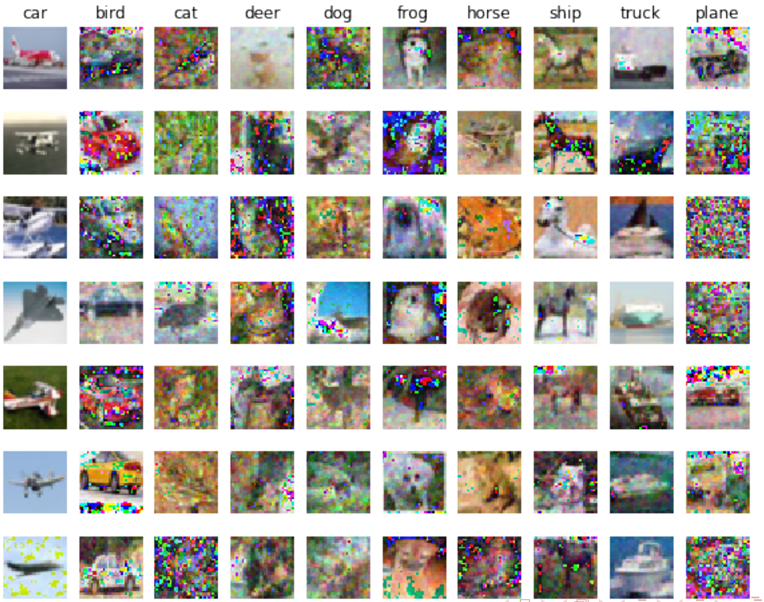
Results of the DRO Learning: Original



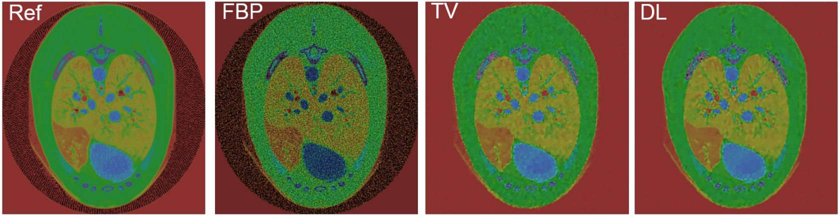
Results of the DRO Learning: Nonrobust



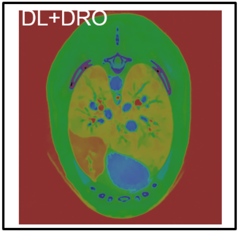
Results of the DRO Learning: DRO



Medical Application



Substantial noise reduction



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]

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.

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

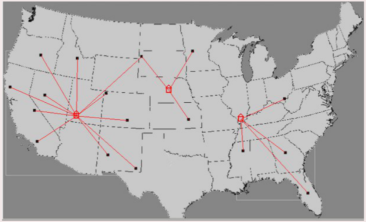
- The DRO models yield a solution with a guaranteed confidence level to the possible distributions. Specifically, the confidence region of the distributions can be constructed upon the historical data and sample distributions.
- The DRO models are tractable, and sometimes maintain the same computational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

Table of Contents

Planning under High-Dimensional Stochastic Data

Name	Symbol	Shares	Cost basis	Mkt value	Gain	Gain %
Apple Inc.	AAPL	1	122.4	86.29	-36.11	-29.5
Amazon.com, Inc.	AMZN	2	128.18	101.52	-26.66	-20.8
eBay Inc.	EBAY	1.25	32.23	17.45	-14.78	-45.85
Microsoft Corporation	MSFT	3.02	84.17	58.41	-25.76	-30.61
Google Inc.	GOOG	4	1733.4	1212.44	-521	-30.05
Yahoo! Inc.	YHOO	2	58.06	23.94	-34.12	-58.77
News Corporation	NWS	5	93.5	46.85	-46.65	-49.89
Time Warner Inc.	TWX	4	59.44	40.28	-19.16	-32.23
The Walt Disney Company	DIS	3	92.28	67.44	-24.84	-26.92
Bank of America Corporation	BAC	1	36.74	13.24	-23.5	-63.96
Best Buy Co., Inc.	BBY	1	39.87	27.75	-12.12	-30.4
Comcast Corporation	CMCSK	1	19.41	15.79	-3.62	-18.65
AT&T Inc.	T	2	70.02	56.46	-13.56	-19.37
Netflix, Inc.	NFLX	1	31.26	28.66	-2.6	-8.32
JPMorgan Chase & Co.	JPM	1	37.56	31.01	-6.55	-17.44
Dell Inc.	DELL	1	19.35	10.23	-9.12	-47.13
DivX, Inc.	DIVX	1	9.38	5.1	-4.28	-45.63
Wal-Mart Stores, Inc.	WMT	1	49.9	55.05	5.15	10.32
IAC/InterActiveCorp	IACI	1	19.53	15.77	-3.76	-19.25
Target Corporation	TGT	1	52	33.55	-18.45	-35.48
Adobe Systems Incorporated	ADBE	1	32.55	21.05	-11.5	-35.33
Electronic Arts Inc.	ERTS	1	46.31	15.34	-30.97	-66.88
Monster Worldwide, Inc.	MWW	1	24.75	11.55	-13.2	-53.33
Circuit City Stores, Inc.	CCTYQ	1	3.84	0.14	-3.71	-96.48
Gannett Co., Inc.	GCI	0.41	11.85	3.27	-8.58	-72.42
The McClatchy Company	MNI	0.14	1.24	0.11	-1.13	-91.21
					-910.5	-1035.58

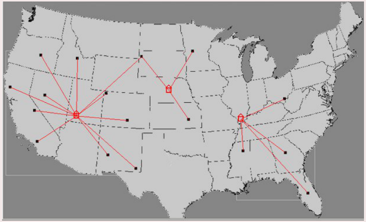
Portfolio Optimization



Facility Location

Planning under High-Dimensional Stochastic Data

Name	Symbol	Shares	Cost basis	Mkt value	Gain	Gain %
Apple Inc.	AAPL	1	122.4	86.29	-36.11	-29.5
Amazon.com, Inc.	AMZN	2	128.18	101.52	-26.66	-20.8
eBay Inc.	EBAY	1.25	32.23	17.45	-14.78	-45.85
Microsoft Corporation	MSFT	3.02	84.17	58.41	-25.76	-30.61
Google Inc.	GOOG	4	1733.4	1212.44	-521	-30.05
Yahoo! Inc.	YHOO	2	58.06	23.94	-34.12	-58.77
News Corporation	NWS	5	93.5	46.85	-46.65	-49.89
Time Warner Inc.	TWX	4	59.44	40.28	-19.16	-32.23
The Walt Disney Company	DIS	3	92.28	67.44	-24.84	-26.92
Bank of America Corporation	BAC	1	36.74	13.24	-23.5	-63.96
Best Buy Co., Inc.	BBY	1	39.87	27.75	-12.12	-30.4
Comcast Corporation	CMCSK	1	19.41	15.79	-3.62	-18.65
AT&T Inc.	T	2	70.02	56.46	-13.56	-19.37
Netflix, Inc.	NFLX	1	31.26	28.66	-2.6	-8.32
JPMorgan Chase & Co.	JPM	1	37.56	31.01	-6.55	-17.44
Dell Inc.	DELL	1	19.35	10.23	-9.12	-47.13
DivX, Inc.	DIVX	1	9.38	5.1	-4.28	-45.63
Wal-Mart Stores, Inc.	WMT	1	49.9	55.05	5.15	10.32
IAC/InterActiveCorp	IACI	1	19.53	15.77	-3.76	-19.25
Target Corporation	TGT	1	52	33.55	-18.45	-35.48
Adobe Systems Incorporated	ADBE	1	32.55	21.05	-11.5	-35.33
Electronic Arts Inc.	ERTS	1	46.31	15.34	-30.97	-66.88
Monster Worldwide, Inc.	MWW	1	24.75	11.55	-13.2	-53.33
Circuit City Stores, Inc.	CCTYQ	1	3.84	0.14	-3.71	-96.48
Gannett Co., Inc.	GCI	0.41	11.85	3.27	-8.58	-72.42
The McClatchy Company	MNI	0.14	1.24	0.11	-1.13	-91.21
					-910.5	-1035.58



Portfolio Optimization **Facility Location**

$$\underset{x \in X}{\text{minimize}} \quad \mathbb{E}_p[f(x,\xi)]$$

where ξ 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:

$$\underset{\mathbf{x}\in X}{\text{minimize}}\ \underset{p\in\mathcal{D}}{\text{maximize}}\ \mathbb{E}_p[f(\mathbf{x},\xi)]$$

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

Price of Correlations

One can also consider the distributionally robust approach:

$$\underset{\mathbf{x}\in X}{\text{minimize}} \ \underset{p\in\mathcal{D}}{\text{maximize}} \ \mathbb{E}_p[f(\mathbf{x},\xi)]$$

where $\mathcal{D}$ is the set of joint distributions such that the marginal distribution of ξ_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:

$$\underset{\mathbf{x}\in X}{\text{minimize}} \ \underset{p\in\mathcal{D}}{\text{maximize}} \ \mathbb{E}_p[f(\mathbf{x},\xi)]$$

where $\mathcal{D}$ is the set of joint distributions such that the marginal distribution of ξ_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

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

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

- $\mathbf{x}^*$ be the optimal solution for the distributionally robust model.

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

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

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

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)]

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

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

Submodularity Leads to Small POC

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

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

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

Submodularity Leads to Small POC

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

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

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

Theorem

If $f(\cdot, \xi)$ is monotone and submodular in ξ, 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 ξ
- 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 $\mathrm{POC} = \Omega(2^n)$.

Agrawal, Ding, Saberi, Y, [2010]

Applications: Stochastic Bottleneck Matching

$$\boxed{\text{minimize}_{\mathbf{x}\in X} \;\; \mathrm{maximize}_{p\in\mathcal{P}} \;\; \mathbb{E}_p[\max_i \xi_i x_i]} \Rightarrow$$

$$\boxed{\mathrm{minimize}_{\mathbf{x}\in X} \;\; \mathbb{E}_{\hat{p}}[\max_i \xi_i x_i]}$$

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

$$\mathrm{minimize}_{\mathbf{x}\in X} \sum_{s\in\mathcal{S}} \max_i\{s_i x_i\}$$

Applications: Stochastic Bottleneck Matching

$$\boxed{\text{minimize}_{\mathbf{x}\in X} \ \ \text{maximize}_{p\in\mathcal{P}} \ \ \mathbb{E}_p[\max_i \xi_i x_i]} \Rightarrow$$

$$\boxed{\text{minimize}_{\mathbf{x}\in X} \ \ \mathbb{E}_{\hat{p}}[\max_i \xi_i x_i]}$$

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

$$\text{minimize}_{\mathbf{x}\in X} \sum_{s\in\mathcal{S}} \max_i\{s_i x_i\}$$

$$\boxed{\text{minimize}_{\mathbf{x}\in X} \ \ \text{maximize}_{p\in\mathcal{P}} \ \ \mathbb{E}_p[||\xi.^*\mathbf{x}||_q]} \Rightarrow$$

$$\boxed{\text{minimize}_{\mathbf{x}\in X} \ \ \mathbb{E}_{\hat{p}}[||\xi.^*\mathbf{x}||_q]}$$

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

- Monotone submodular function, $e/(e-1) \sim 1.6$ approximation.

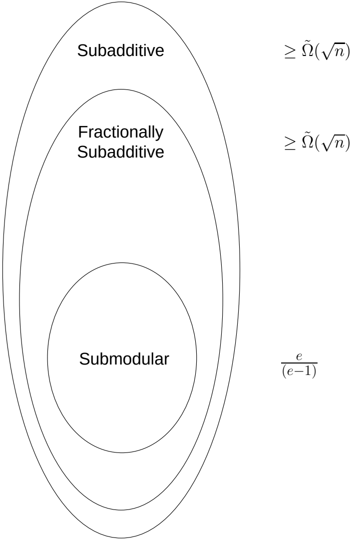
Beyond Submodularity?

Monotone Subadditive Functions?

- Preserves economy of scale
- Example with POC $\geq \Omega(\sqrt{n}/\log\log(n))$

Fractionally Subadditive?

- POC $\geq \Omega(\sqrt{n}/\log\log(n))$



Beyond Submodularity?

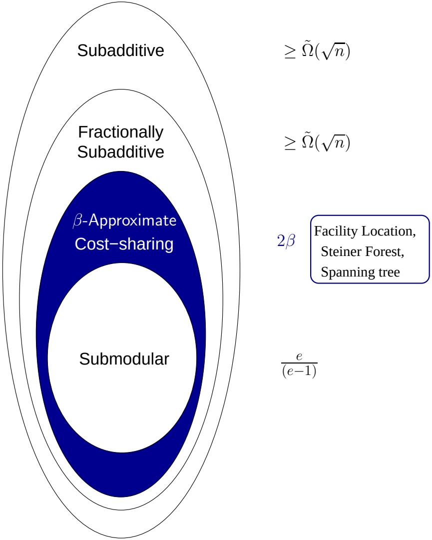
Monotone Subadditive Functions?

- Preserves economy of scale
- Example with $\text{POC} = \Omega(\sqrt{n}/\log\log(n))$

Fractionally Subadditive?

- $\text{POC} \geq \Omega(\sqrt{n}/\log\log(n))$

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.
- β-approximate cost-sharing scheme: total cost charged is within β 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?
  [introduced by Thomson (1983, 1995) in context of bargaining]
- For submodular functions – charge marginal costs.
- β-approximate cost-sharing scheme: total cost charged is within β 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 ξ with β-cost-sharing scheme, POC $\leq 2\beta$.

- POC ≤ 6 for two-stage stochastic facility location
- POC ≤ 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 β, there exists a cross-monotone cost-sharing scheme with expected β-budget balance.

We show examples of

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

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

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.

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.

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).

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 $i \in \{1, ..., m\}$ 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 $i \in \{1, ..., m\}$ 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 $\mathbf{p}_j$.

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 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 $\mathbf{p}_j$.
- 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.

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 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 $\mathbf{p}_j$.
- 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.

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 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 $\mathbf{p}_j$.
- 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.

The Optimal Cost-to-Go Value Vector I

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

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*,\ \forall j \in \mathcal{A}_i\},\ \forall i,$$

with optimal policy

$$\pi_i^* = \arg\min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*,\ \forall j \in \mathcal{A}_i\},\ \forall i.$$

The Optimal Cost-to-Go Value Vector I

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

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i,$$

with optimal policy

$$\pi_i^* = \arg\min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i.$$

In the Game setting, the conditions becomes:

$$y_i^* = \min\{c_j + \gamma \mathbf{p}_j^T \mathbf{y}^*, \ \forall j \in \mathcal{A}_i\}, \ \forall i \in I^-,$$

and

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

They both are fix-point or saddle-point problems.

Value-Iteration (VI) Method

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

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

Value-Iteration (VI) Method

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

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

The VI for MGP

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

and

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

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

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

Randomized/Sample-Based Value-Iteration

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T \mathbf{y}^k$ by stochastic sampling,

Randomized/Sample-Based Value-Iteration

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T\mathbf{y}^k$ by stochastic sampling,
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T\mathbf{y}^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.

Randomized/Sample-Based Value-Iteration

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T \mathbf{y}^k$ by stochastic sampling,
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T \mathbf{y}^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.
- Since randomization is introduced in the algorithm, the iterative solution sequence becomes a random sequence.

Randomized/Sample-Based Value-Iteration

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T\mathbf{y}^k$ by stochastic sampling,
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T\mathbf{y}^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.
- Since randomization is introduced in the algorithm, the iterative solution sequence becomes a random sequence.
- 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.

Randomized/Sample-Based Value-Iteration

- In many practical applications, $\mathbf{p}_j$ is unknown so that we have to approximate the mean $\mathbf{p}_j^T \mathbf{y}^k$ by stochastic sampling,
- Even we know $\mathbf{p}_j$ exactly, it may be too dense so that the computation of $\mathbf{p}_j^T \mathbf{y}^k$ takes up to $O(m)$ operations so that we would rather estimate the mean by sampling which can be easily parallelized.
- Since randomization is introduced in the algorithm, the iterative solution sequence becomes a random sequence.
- 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.

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):

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 $\mathbf{p}_j$:

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

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

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 $\mathbf{p}_j$:

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

  to compute an ϵ-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 ϵ-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 $\mathbf{p}_j$:

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

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

- Sampling on the pure generative model:

$$O\left(\frac{n}{(1-\gamma)^4\epsilon^2}\log(\frac{1}{\delta})\right)$$

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

- Sample complexity lower bound: $O\left(\frac{n}{(1-\gamma)^3\epsilon^2}\right)$.

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 ϵ-optimal policy with probability at least $1-\delta$.

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 ϵ-optimal policy with probability at least $1-\delta$.
- The method is also extended to computing ϵ-optimal policies for finite-horizon MDP with a generative model and provide a nearly matching sample complexity lower bound.

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 ϵ-optimal policy with probability at least $1-\delta$.
- The method is also extended to computing ϵ-optimal policies for finite-horizon MDP with a generative model and provide a nearly matching sample complexity lower bound.
- More recently, the result is substantially generalized to the Stochastic Game process.

Table of Contents

Background

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

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
- Decision: To sell or not to each individual customer?

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
- Decision: To sell or not to each individual customer?
- Objective: Maximize the revenue.

An Example

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

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

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

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

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

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

- We only know $\mathbf{b}$ 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, $t = 1, ..., n$ and

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

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

- We only know $\mathbf{b}$ 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

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

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

- We only know $\mathbf{b}$ 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 $(\mathbf{a}_t, \pi_t)$ 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 $(\mathbf{a}_t, \pi_t)$ arrive in a random order (rather than from some iid distribution).

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

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

where σ 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

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

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

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

Agrawal, Wang and Y [2010, 2014]

Ideas to Prove Negative Result

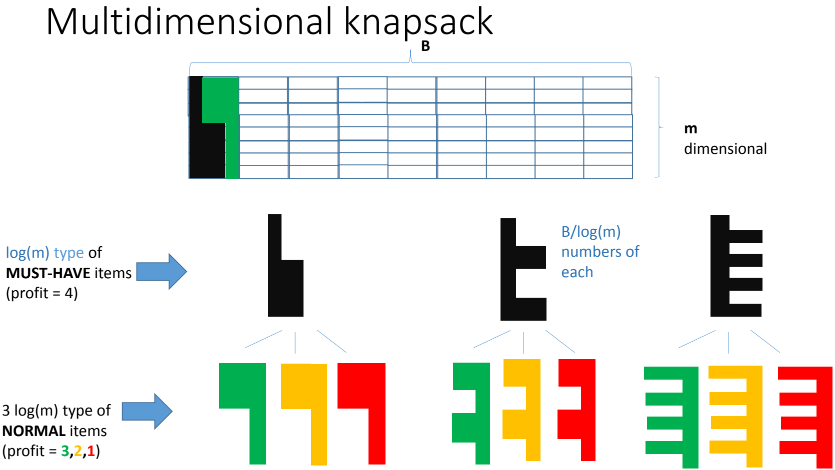
- Consider $m = 1$ and inventory level B, one can construct an instance where $OPT = B$, and there will be a loss of $\sqrt{B}$ with a high probability, which give an approximation ratio $1 - \frac{1}{\sqrt{B}}$.

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 $OPT = B/\log(m)$.

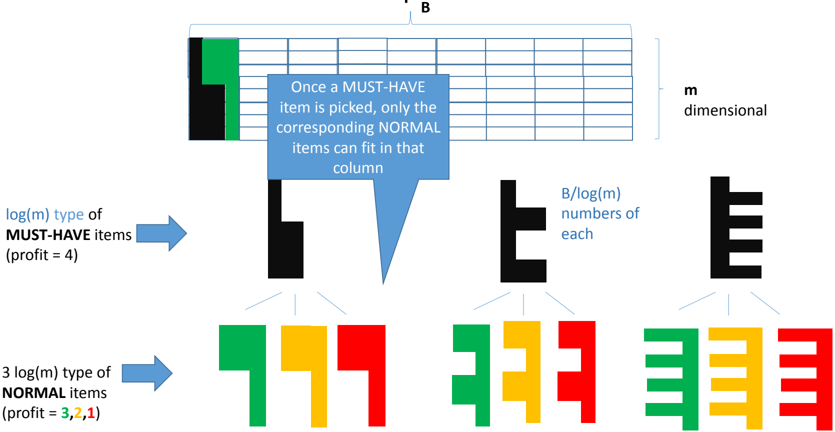
Ideas to Prove Negative Result

- 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 $OPT = B/\log(m)$.
- Then, with hight 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 $1 - \frac{\sqrt{\log(m)}}{\sqrt{B}}$.


Multidimensional knapsack
B
m
dimensional
log(m) type of
MUST-HAVE items
(profit = 4)
B/log(m)
numbers of
each
3 log(m) type of
NORMAL items
(profit = 3,2,1)

Necessary Result II

Multidimensional knapsack



Ideas to Prove Positive Result: Dynamic Learning

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 can not be carried out to find an optimal sequential decision policy.
- Thus, the online algorithm needs to be learning-based, in particular, learning-while-doing.

Price Observation of Online Learning I

The problem would be easy if there are "ideal prices":

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

Price Observation of Online Learning II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.

Price Observation of Online Learning II

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

Price Observation of Online Learning II

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

Price Observation of Online Learning II

- Pricing the bid: The optimal dual price vector $\mathbf{p}^*$ of the offline LP problem can play such a role, that is $x_t^* = 1$ if $\pi_t > \mathbf{a}_t^T \mathbf{p}^*$ and $x_t^* = 0$ otherwise, yields a near-optimal solution.
- Based on this observation, our online algorithm works by learning a threshold price vector $\hat{\mathbf{p}}$ and using $\hat{\mathbf{p}}$ to price the bids.
- One-time learning algorithm: learn the price vector once using the initial ϵn input.
- Dynamic learning algorithm: dynamically update the prices at a carefully chosen pace.

One-Time Learning Algorithm

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 ϵ portion of the problem

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

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

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 ϵ portion of the problem

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

  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 $x_t = 0$.

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 ϵ 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 ϵ portion revenue;
- With high probability, the first ϵ 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 ϵ portion revenue;
- With high probability, the first ϵ 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 ϵ factor worse than the lower bound...

Dynamic Learning Algorithm

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

Dynamic Learning Algorithm

In the dynamic price learning algorithm, we update the price at time ϵn, $2\epsilon n$, $4\epsilon n$, ..., till $2^k\epsilon \geq 1$.
At time $\ell \in \{\epsilon n, 2\epsilon n, \ldots\}$, the price vector is the optimal dual solution to the following linear program:

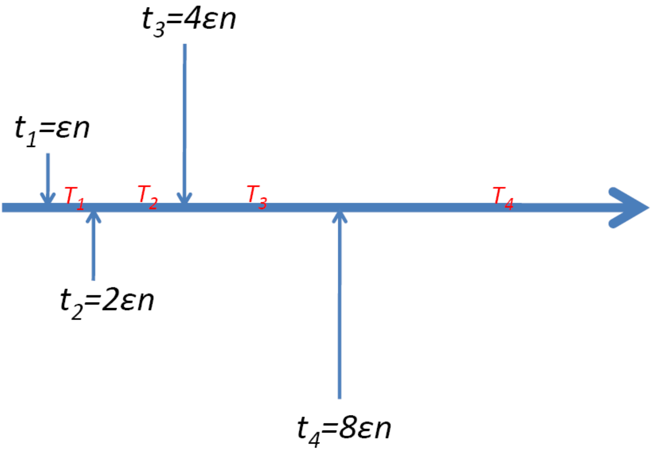
$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1-h_\ell)\frac{\ell}{n} b_i & i = 1, \ldots, m \\ & 0 \leq x_t \leq 1 & t = 1, \ldots, \ell \end{array}$$

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



Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon.

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon.
- The numbers h_ℓ 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 $\log_2(1/\epsilon)$ times during the entire time horizon.
- The numbers h_ℓ 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_ℓ (more conservative) at the beginning periods and smaller h_ℓ (more aggressive) at the later periods, one can now control the loss of revenue by an ϵ order while the required size of B can be weakened by an ϵ factor.

Related Work on Random-Permutation

	Sufficient Condition	Learning
Kleinberg [2005]	$B \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic

Related Work on Random-Permutation

	Sufficient Condition	Learning
Kleinberg [2005]	$B \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic
Devanur et al [2009]	$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$	One-time

Related Work on Random-Permutation

	Sufficient Condition	Learning
Kleinberg [2005]	$B \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic
Devanur et al [2009]	$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$	One-time
Feldman et al [2010]	$B \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$	One-time

Related Work on Random-Permutation

	Sufficient Condition	Learning
Kleinberg [2005]	$B \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic
Devanur et al [2009]	$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$	One-time
Feldman et al [2010]	$B \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$	One-time
Agrawal et al [2010]	$B \geq \frac{m \log n}{\epsilon^2}$ **or** $OPT \geq \frac{m^2 \log n}{\epsilon^2}$	Dynamic

Related Work on Random-Permutation

	Sufficient Condition	Learning
Kleinberg [2005]	$B \geq \frac{1}{\epsilon^2}$, for $m = 1$	Dynamic
Devanur et al [2009]	$OPT \geq \frac{m^2 \log(n)}{\epsilon^3}$	One-time
Feldman et al [2010]	$B \geq \frac{m \log n}{\epsilon^3}$ **and** $OPT \geq \frac{m \log n}{\epsilon}$	One-time
Agrawal et al [2010]	$B \geq \frac{m \log n}{\epsilon^2}$ **or** $OPT \geq \frac{m^2 \log n}{\epsilon^2}$	Dynamic
Molinaro/Ravi [2013]	$B \geq \frac{m^2 \log m}{\epsilon^2}$	Dynamic

Related Work on Random-Permutation

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

Table: Comparison of several existing results

Price-Post Learning

- Selling a good in a fixed horizon T, and there is no salvage value for the remaining quantities after the horizon.

Price-Post Learning

- 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.

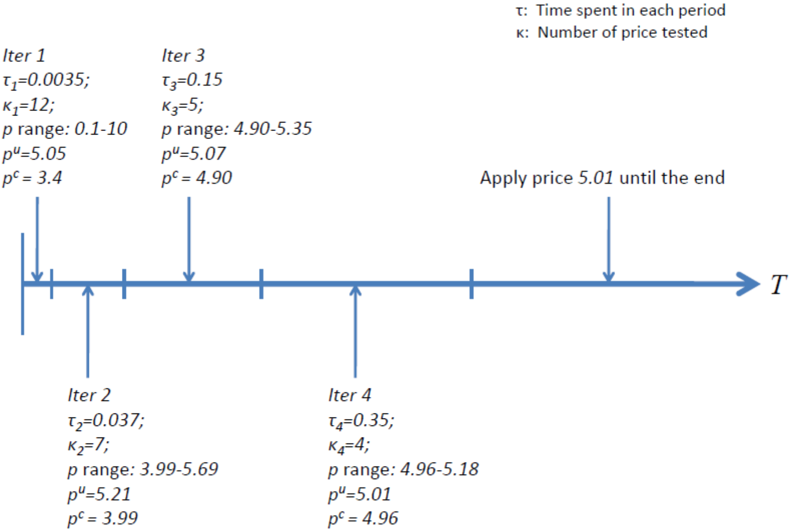
Price-Post Learning

- 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.

Price-Post Learning

- 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



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).

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.

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?