Distributionally Robust Optimization, Online Linear Programming and Markets for Public-Good Allocations

Models/Algorithms for Learning and Decision Making Driven by Data/Samples

Yinyu Ye

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

Mathematical Foundations of Data Science, June 2, 2020
(Joint work with many others ...)

Outline

- Introduction to Stochastic and Distributionally Robust Optimization (DRO)
- DRO under Moment, Likelihood and Wasserstein Bounds (Probability Theory)

Outline

- Introduction to Stochastic and Distributionally Robust Optimization (DRO)
- DRO under Moment, Likelihood and Wasserstein Bounds (Probability Theory)
- Online (Binary) Linear Optimization and Dynamic Resource Allocation (Optimization Theory)

Outline

- Introduction to Stochastic and Distributionally Robust Optimization (DRO)
- DRO under Moment, Likelihood and Wasserstein Bounds (Probability Theory)
- Online (Binary) Linear Optimization and Dynamic Resource Allocation (Optimization Theory)
- Markets for Efficient Public Good Allocation (Economic Theory)

Outline

- Introduction to Stochastic and Distributionally Robust Optimization (DRO)
- DRO under Moment, Likelihood and Wasserstein Bounds (Probability Theory)
- Online (Binary) Linear Optimization and Dynamic Resource Allocation (Optimization Theory)
- Markets for Efficient Public Good Allocation (Economic Theory)

Develop tractable and provable models and algorithms for decision-making and optimization with uncertain, online/dynamic and massive data.

Table of Contents

Introduction to SO and 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 $\mathbf{x}$ (typically called β in data science with minimizing a loss function) are decision variables with feasible region X, ξ represents random variables satisfying joint distribution F_ξ.

Introduction to SO and DRO

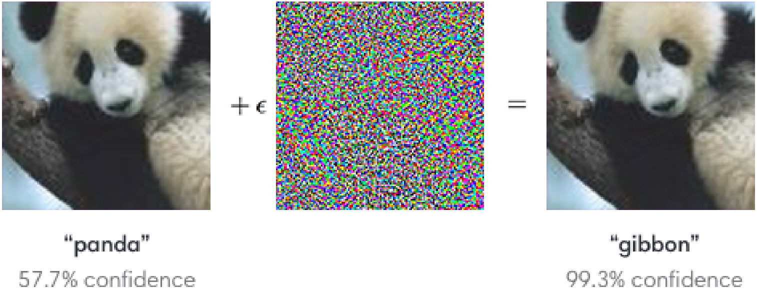
We start from considering a stochastic optimization problem as follows:

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

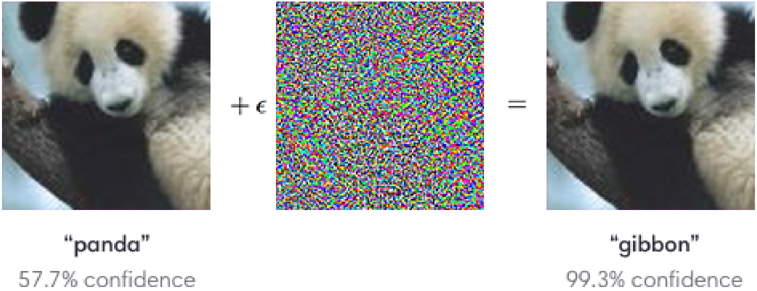
where $\mathbf{x}$ (typically called β in data science with minimizing a loss function) are decision variables with feasible region X, ξ represents random variables satisfying joint distribution F_ξ.

- Pros: In many cases, the expected value is a good measure of performance; simply apply simple sample average (SSA) approach.
- 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 and it ignores observed/training data information/statistics. 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 (fast algorithm available)

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 (fast algorithm available)
- Practical (Statistical) Meanings (utilize observed/training data)

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 (fast algorithm available)
- Practical (Statistical) Meanings (utilize observed/training data)
- Performance (the potential loss comparing to the benchmark cases)

Brief 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 Ḿohring et al. [1999] considers the product distribution

Table of Contents

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, where μ_0 and Σ_0 are sample mean vector and variance matrix.

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, where μ_0 and Σ_0 are sample mean vector and variance matrix.

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

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

where again 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 $||\xi||_2 \le R$ a.s..

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

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

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 the likelihood/entropy function

$$L(\xi, F_\xi) = \sum_{i=1}^{n} \frac{m_i}{n} \log p_i \quad \text{or} \quad L(\xi, F_\xi) = \sum_{i=1}^{n} -p_i \log(p_i \frac{n}{m_i}).$$

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 [12,16]

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

$$\mathcal{D}_N := \{F_\xi \mid P(\xi \in \Xi) = 1 \;\&\; d(F_\xi, \hat{F}_N) \le \varepsilon_N\},$$

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 with Wasserstein Ambiguity 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 with Wasserstein Ambiguity 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 with Wasserstein Ambiguity 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 :

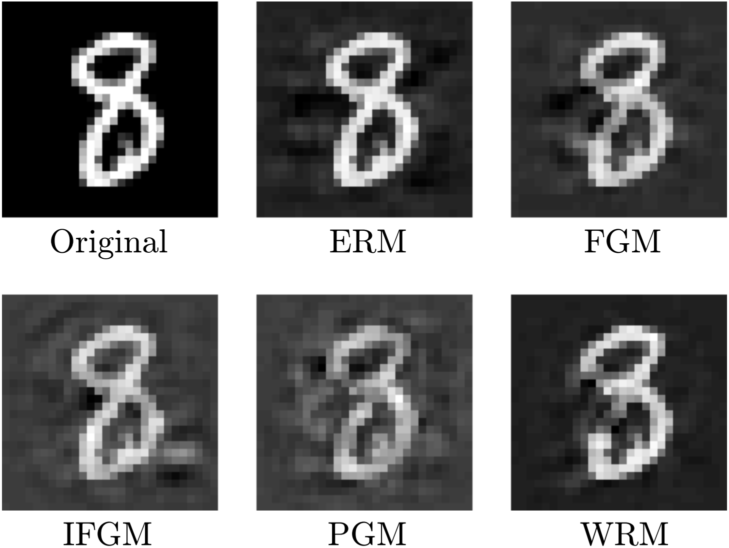
$$\min_x \frac{1}{N} \sum_{i=1}^N \ell(x, \hat{\xi}_i, \hat{\lambda}_i) \text{ where } \ell(x, \xi, \lambda) = \ln(1 + \exp(-\lambda x^T \xi))$$

- 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:

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

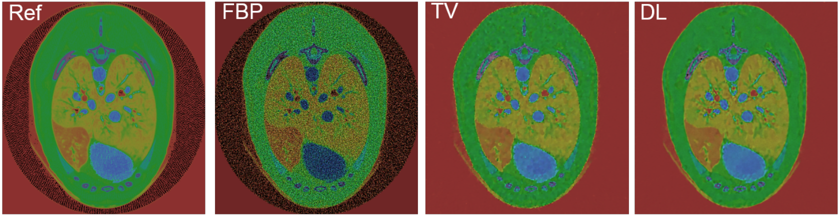
Shafieezadeh Abadeh, Mohajerin Esfahani, & Kuhn, NIPS, [2015]

Results of the DRO Learning

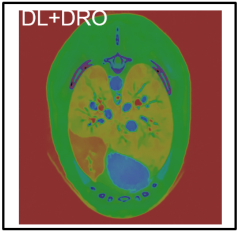


Sinha, Namkoong and Duchi [2017]

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]

Distributionally Robust Non-parametric Conditional Estimation

Conditional estimation given specific covariate values (i.e., local conditional estimation or functional estimation) is ubiquitously useful with applications in engineering, social and natural sciences. We may

$$\min_{\psi} \ \mathbb{E}[\| Y - \psi(X)\|_2^2],$$

where the min is taken over all infinite measurable functions.

Or it can be cast into solving a family of finite-dimensional optimization problems parametrically in $\mathbf{x}_0$:

$$\min_{\beta} \mathbb{E}_F[\ell(Y, \beta)|X = \mathbf{x}_0] \tag{4}$$

with an appropriately chosen statistical loss function ℓ and a response variable Y, given the value or observation over a covariate X.

Local Conditional Estimator

But it may happen that there is no or little training data with the exact covariate $X = \mathbf{x}_0$. Then one can consider solving a Local conditional estimation problem:

$$\min_{\beta} \mathbb{E}_F[\ell(Y, \beta) | X \in N_\gamma(x_0)] \tag{5}$$

where the expectation is conditioned on a neighborhood $N_\gamma(\mathbf{x}_0)$ around $\mathbf{x}_0$ of radius γ. Under reasonable regularity assumptions, problem (4) can be viewed as the limit of (5) as the radius γ of the neighborhood shrinks to zero.

There are always added errors which may be due to finite sampling, noise and corrupted data. This motivates us to investigate potential strategies to robustify the local conditional estimation problem (5) under the framework of distributionally robust optimization.

Performance (Nguyen et al. 2020)

We compare to DRLE k-nearest neighbour (KNN), Nadaraya-Watson (N-W), and Nadaraya-Epanechnikov (N-E) estimators on a digit estimation problem using the MNIST database. We executed 100 experiments where training and test sets were randomly drawn without replacement from the 60,000 training examples of this dataset.

Method	$N = 50$	$N = 100$	$N = 500$
KNN	$25\% \pm 2\%$	$30\% \pm 2\%$	$59\% \pm 2\%$
N-W	$29\% \pm 2\%$	$39\% \pm 2\%$	$63\% \pm 1\%$
N-E	$26\% \pm 1\%$	$34\% \pm 1\%$	$48\% \pm 1\%$
DRLE	$34\% \pm 2\%$	$47\% \pm 2\%$	$70\% \pm 1\%$

Table: Comparison of expected out-of-sample classification accuracy when rounding the estimators' outputs under different training set sizes. The 90% confidence interval are estimated based on 100 experiments.

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

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($\mathbf{b}$)
Reward(r_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

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

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

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

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

- We only know $\mathbf{b}$ and n at the start

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

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

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.

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

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

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.
- 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);
- $r_t \geq 0$ for all t
- The bids $(\mathbf{a}_t, r_t)$ 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);
- $r_t \geq 0$ for all t
- The bids $(\mathbf{a}_t, r_t)$ arrive in a random order (rather than from some iid distribution).

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

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

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

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

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]

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(r_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 $r_t > \mathbf{a}_t^T \mathbf{p}^*$ 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 $r_t > \mathbf{a}_t^T \mathbf{p}^*$ 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 $r_t > \mathbf{a}_t^T\mathbf{p}^*$ 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 $r_t > \mathbf{a}_t^T \mathbf{p}^*$ 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

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

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 $1 \le t \le \epsilon n$;
- Solve the ϵ portion of the problem

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

  and get the optimal dual solution $\hat{\mathbf{p}}$;
- Determine the future allocation x_t as:

$$x_t = \begin{cases} 0 & \text{if } r_t \le \hat{\mathbf{p}}^T \mathbf{a}_t \\ 1 & \text{if } r_t > \hat{\mathbf{p}}^T \mathbf{a}_t \end{cases}$$

  as long as $a_{it} x_t \le b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ 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 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 $\ell \in \{\epsilon n, 2\epsilon n, \dots\}$, the price vector is the optimal dual solution to the following linear program:

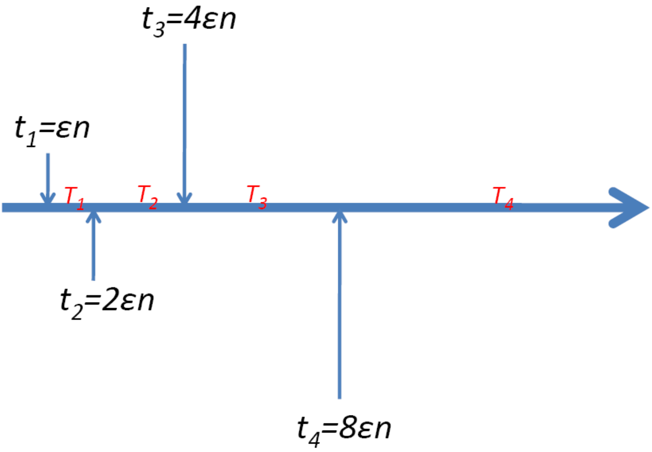
$$
\begin{array}{lll}
\text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} r_t x_t & \\
\text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1 - h_\ell)\frac{\ell}{n} b_i & i = 1, \dots, m \\
 & 0 \leq x_t \leq 1 & t = 1, \dots, \ell
\end{array}
$$

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. One can also update prices or resolve an LP problem after every bid to achieve a lightly better competitive ratio.

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon. One can also update prices or resolve an LP problem after every bid to achieve a lightly better competitive ratio.
- 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 possible left-over goods.

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon. One can also update prices or resolve an LP problem after every bid to achieve a lightly better competitive ratio.
- 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 possible left-over goods.
- Does the price vector converges?

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon. One can also update prices or resolve an LP problem after every bid to achieve a lightly better competitive ratio.
- 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 possible left-over goods.
- Does the price vector converges?
- Can the model handle the double-market?

Comments on Dynamic Learning Algorithm

- In the dynamic algorithm, we update the prices $\log_2(1/\epsilon)$ times during the entire time horizon. One can also update prices or resolve an LP problem after every bid to achieve a lightly better competitive ratio.
- 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 possible left-over goods.
- Does the price vector converges?
- Can the model handle the double-market?
- Could the online algorithm avoid solving LPs?

Recent Results: Dual Convergence I

The offline dual problem again

$$\begin{array}{rc} \min & \sum_{i=1}^m b_i p_i + \sum_{j=1}^n y_j \\ \text{s.t.} & \sum_{i=1}^m a_{ij} p_i + y_j \geq r_j, \quad j = 1, ..., n. \\ & p_i, y_j \geq 0 \text{ for all } i, j. \end{array}$$

can be rewritten, with $\mathbf{d} = \mathbf{b}/n$, as:

$$\begin{array}{rc} \min & \sum_{i=1}^m d_i p_i + \sum_{j=1}^n \left(r_j - \sum_{i=1}^m a_{ij} p_i\right)^+ \\ \text{s.t.} & p_i \geq 0, \quad i = 1, ..., m. \end{array}$$

Here, entries of $(r_j, \mathbf{a}_j)$ can be either positive or negative!

Recent Results: Dual Convergence II

Normalize the objective,

$$\begin{array}{lll} \min & f_n(\mathbf{p}) := & \sum_{i=1}^m d_i p_i + \frac{1}{n} \sum_{j=1}^n \left(r_j - \sum_{i=1}^m a_{ij} p_i\right)^+ \\ \text{s.t.} & & p_i \geq 0, \quad i = 1, ..., m. \end{array}$$

Define the stochastic program

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

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

The optimal dual solution $\mathbf{p}_n^*$ converges to the (unique) optimal solution $\mathbf{p}^*$ of the stochastic program. But at what rate?

Recent Results: Dual Convergence III

Theorem

(Li and Y 2019) Define the binding index set $B = \{i : p_i^* = 0\}$ *and the non-binding index set* $N = \{i : p_i^* \neq 0\}$. *Under standard stochastic assumptions (iid...), there exists a universal constant* C *such that*

$$\mathbb{P}\left(\|\mathbf{p}_n^*(B) - \mathbf{p}^*(B)\|_2 \geq \epsilon\right) \leq m \exp\left(-\frac{Cn\epsilon^2}{m}\right)$$

$$\mathbb{P}\left(\|\mathbf{p}_n^*(N) - \mathbf{p}^*(N)\|_2 \geq \epsilon\right) \leq m \exp\left(-\frac{Cn\epsilon}{m}\right)$$

for any $\epsilon > 0$, $m, n \in \mathbb{N}^+$ *and* $\mathcal{P} \in \Xi$. *Here* $\mathbf{p}_n^*(B)$, $\mathbf{p}^*(B)$, $\mathbf{p}_n^*(N)$, $\mathbf{p}^*(N)$ *denote the sub-vectors corresponding to the indexes in* B, N.

Performance Metrics

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

$$R_n^* = \sum_{j=1}^{n} r_j x_j^* \quad \text{and} \quad R_n = \sum_{j=1}^{n} r_j x_j.$$

Objective: Minimize the worst-case gap, Regret, between the offline and online objective values

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

Also, we measure the constraint violation of the online solution

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

Remark: A bi-objective performance measure

Recent Results: Action-Dependent Learning I

Earlier Online Algorithms

At each time t, compute the dual optimal solution for the problem:

$$\begin{aligned} \max \ & \sum_{j=1}^{t-1} r_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{t-1} a_{ij} x_j \leq b_i(t-1)/n, \quad \forall i \\ & 0 \leq x_j \leq 1, \quad j = 1, ..., t-1 \end{aligned}$$

The algorithms are estimating $\mathbf{p}^*$ purely using data.

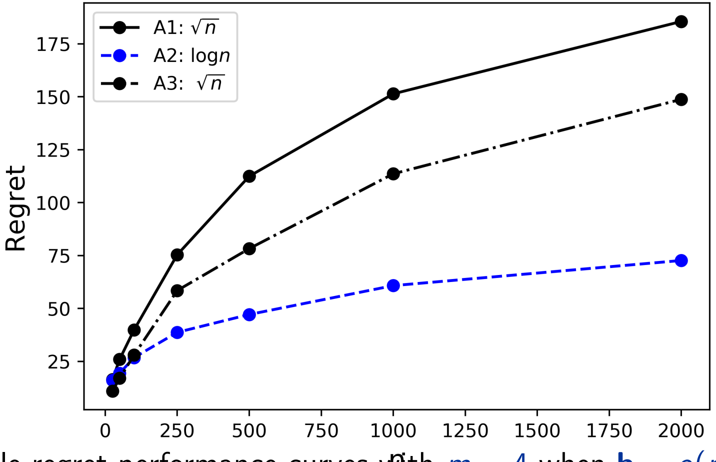
Action-Dependent Learning

At each time t, compute the dual optimal solution for the problem:

$$\begin{aligned} \max \ & \sum_{j=1}^{t-1} r_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{t-1} a_{ij} x_j \leq \frac{\tilde{b}_i(t-1)}{n-t+1}, \ \forall i \\ & 0 \leq x_j \leq 1, \quad j = 1, ..., t-1 \end{aligned}$$

where $\tilde{b}_i = b_i - \sum_{j=1}^{t-1} a_{ij} \tilde{x}_j$ – the new algorithm also considers actions already taken.

Recent Results: Action-Dependent Learning II



Provable regret performance curves with $m = 4$ when $\mathbf{b} = o(n)$, where A1, A2, and A3 stand for Algorithm 1 (Early Dynamic Learning), Algorithm 2 (Action-Dependent), and Algorithm 3 (Using $\mathbf{p}^*$), respectively.

Fast LP-Free Algorithm

Input: $\mathbf{d} = \mathbf{b}/n$ and Initialize $\mathbf{p}^1 = \mathbf{0}$

For $t = 1, ..., n$

Set

$$x_t = \begin{cases} 1, & r_t > \mathbf{a}_t^\top \mathbf{p}_t \\ 0, & r_t \leq \mathbf{a}_t^\top \mathbf{p}_t \end{cases}$$

Compute

$$\begin{aligned} \mathbf{p}_{t+1} &= \mathbf{p}_t + \gamma_t \left(\mathbf{a}_t x_t - \mathbf{d} \right) \\ \mathbf{p}_{t+1} &= \mathbf{p}_{t+1} \vee \mathbf{0} \end{aligned}$$

Output: $\mathbf{x} = (x_1, ..., x_n)$.

Stochastic subgradient descent for the dual equivalent form!

Performance of the Stochastic Input Model

Theorem (Li, Sun, & Ye (2020))

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

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

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

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

Remark: The proof utilizes the structure of the LP and largely mimics the analyses of the online gradient descent algorithm.

Performance of the Random Permutation Model

Theorem (Li, Sun, & Ye (2020))

With the step size $\gamma_t = \frac{1}{\sqrt{n}}$, *the regret and expected constraint violation of the algorithm satisfy*

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

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

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

An extra $\log(n)\sqrt{n}$ in the Regret for the Permutation Input Model compared with that in the Stochastic iid Input Model. The proof builds upon the notion of *Permutational Rademacher Complexity* (Tolstikhin et al. 2015) which is originally used for analyzing transductive learning.

Two LP-Based Dynamic Algorithms

Theorem (Li, Sun, & Ye (2020))

Under the random permutation model, the regret and expected constraint violation of the two algorithms (Agrawal et al. 2014, Kesselheim et al. 2014) both satisfy

$$R_n^* - \mathbb{E}[R_n] \leq O\left(\sqrt{mn}\right)$$

$$\mathbb{E}[v(\mathbf{x})] \leq O(\sqrt{mn}\log n)$$

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

Remark: Compared with the fast algorithm, the regret and constraint violation are reduced by a factor of $\sqrt{m}$ with the price of more computation cost.

Numerical Experiments I

Multi-knapsack instances: Chu and Beasley (1998), Drake et al. (2016).

		Gurobi	Fast Alg.	Alg. 1	Alg. 2
$m = 5, n = 500$	CPU time	0.260	0.039	487	487
	Cmpt. Ratio	100%	94.6%	95.0%	94.9%
$m = 10, n = 500$	CPU time	0.350	0.029	373	373
	Cmpt. Ratio	100%	95.74%	94.09%	93.94%
$m = 30, n = 500$	CPU time	0.310	0.039	491	491
	Cmpt. Ratio	100%	95.6%	93.9%	92.2%

- Alg. 1: Agrawal et al. (2014); Alg. 2: Kesselheim et al. (2014)
- Gurobi computes the optimal solution in an offline fashion while the other three algorithms are online.
- Gurobi is set to solve the binary LP problem. The optimality ratio is reported against the objective value of the Gurobi's binary solution.

Numerical Experiments II

Large-scale multi-knapsack problem:

		Gurobi	Fast Alg.
$m = 100, n = 10,000$	CPU time	34.7	1.04
	Cmpt. Ratio	100%	97.1%
$m = 500, n = 10,000$	CPU time	267	1.05
	Cmpt. Ratio	100%	95.2%
$m = 1000, n = 10,000$	CPU time	764	1.47
	Cmpt. Ratio	100%	94.9%

- The other two algorithms are too slow to finish while Gurobi is set to solve the relaxed LP problem.
- The optimality ratio is reported against the objective value of the relaxed LP, and it decreases as m grows with fixed n.

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.
- Better to do dynamic learning, that is, "learning-while-doing", and the Action-Dependent dynamic learning is even better.

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.
- Better to do dynamic learning, that is, "learning-while-doing", and the Action-Dependent dynamic learning is even better.
- 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.
- Better to do dynamic learning, that is, "learning-while-doing", and the Action-Dependent dynamic learning is even better.
- Multi-item price-posting market?
- More general online optimization?

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.
- Better to do dynamic learning, that is, "learning-while-doing", and the Action-Dependent dynamic learning is even better.
- Multi-item price-posting market?
- More general online optimization?
- Approximately solve large-scale offline binary LPs with the proposed fast algorithm?

Table of Contents

Capacity Constraints on "Public Goods"

Either open: An overcrowded open beach

Capacity Constraints on “Public Goods”

Either open: An overcrowded open beach

Or closed: a completely empty beach generating no value to society

Capacity Constraints on “Public Goods”

Either open: An overcrowded open beach

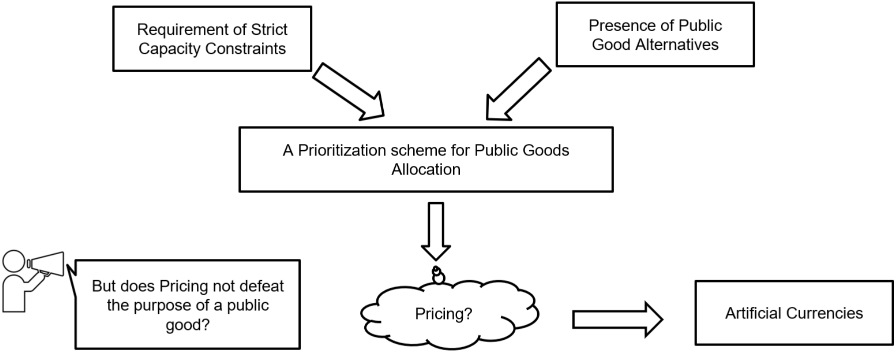
Or closed: a completely empty beach generating no value to society

A pandemic adds capacity restriction to shops, gyms, schools, and public spaces such as parks and beaches, which seems limitless under normal times.

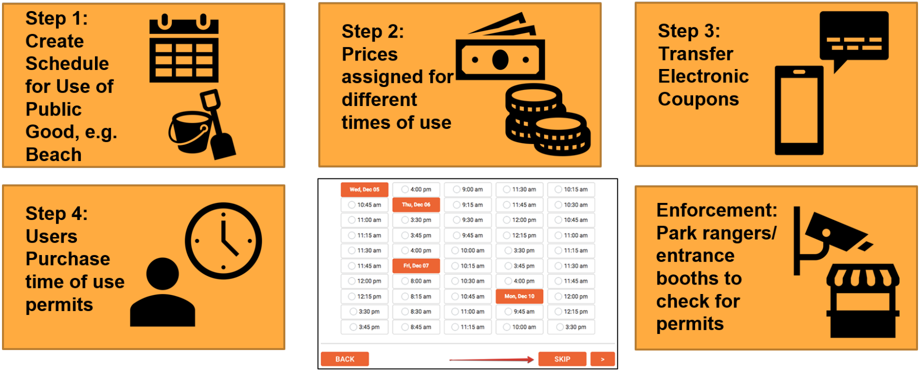
"Time-of-Use" on "Public Goods"

To achieve an intermediate between the two extreme scenarios, open or closed, through Time-of-Use goods. More precisely, create different time periods and people "book/purchase" permits to use the public spaces at one time-period so that the population density on spaces can be upper limited.
(Jalota, Pavone and Y 2020)

Market-Based Mechanism Design



Market-Based Solution



How to setup "prices" for each time-period/good so that resources can be efficiently allocated while keep each individual satisfied?

The Model: Fisher's Equilibrium Price

Buyer $i \in B$'s optimization problem for given prices p_j, $j \in G$.

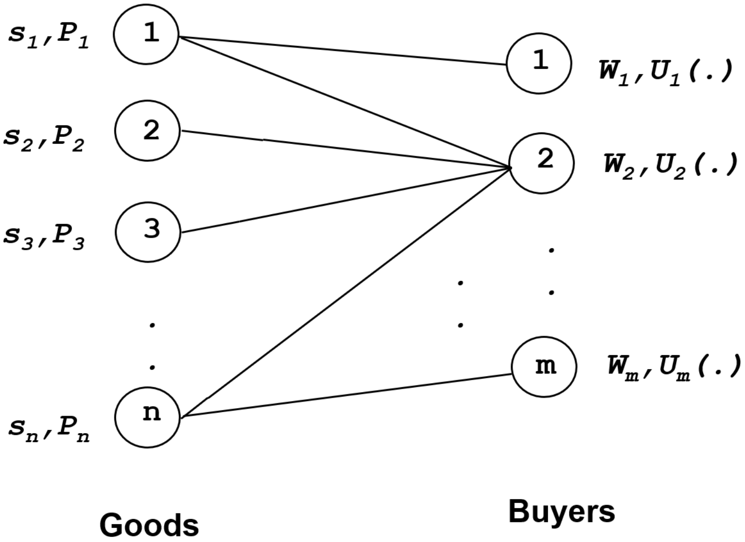
$$\begin{array}{rl} \max & \mathbf{u}_i^T \mathbf{x}_i := \sum_{j \in G} u_{ij} x_{ij} \\ \text{s.t.} & \mathbf{p}^T \mathbf{x}_i := \sum_{j \in G} p_j x_{ij} \leq w_i, \\ & x_{ij} \geq 0, \quad \forall j, \end{array}$$

Assume that the given amount of each good is $\bar{s}_j$. The equilibrium price vector is the one that for all $j \in G$

$$\sum_{i \in B} x^*(\mathbf{p})_{ij} = \bar{s}_j$$

where $\mathbf{x}^*(\mathbf{p})$ is a maximizer of the utility maximization problem for every buyer i.

The Fisher Market Illustration



The Aggregated Social Optimization Problem

$$
\begin{array}{rl}
\max & \sum_{i \in B} w_i \log(\mathbf{u}_i^T \mathbf{x}_i) \\
\text{s.t.} & \sum_{i \in B} x_{ij} = \bar{s}_j, \quad \forall j \in G \\
& x_{ij} \geq 0, \quad \forall i, j,
\end{array}
$$

Theorem

(Eisenberg and Gale 1959) Optimal dual (Lagrange) multiplier vector of equality constraints is an equilibrium price vector.

Proof: The optimality conditions of the social problem are identical to the equilibrium conditions.

The Model with Individual Physical Constraints

Buyer $i \in B$'s optimization problem for given prices p_j, $j \in G$.

$$\begin{array}{rc} \max & \mathbf{u}_i^T \mathbf{x}_i := \sum_{j \in G} u_{ij} x_{ij} \\ \text{s.t.} & \mathbf{p}^T \mathbf{x}_i := \sum_{j \in G} p_j x_{ij} \leq w_i, \\ & A_i \mathbf{x}_i \leq \mathbf{b}_i, \\ & x_{ij} \geq 0, \quad \forall j, \end{array}$$

Could we still solve (?)

$$\begin{array}{rc} \max & \sum_{i \in B} w_i \log(\mathbf{u}_i^T \mathbf{x}_i) \\ \text{s.t.} & \sum_{i \in B} x_{ij} = \bar{s}_j, \quad \forall j \in G \\ & A_i \mathbf{x}_i \leq \mathbf{b}_i, \forall i \in B, \\ & x_{ij} \geq 0, \quad \forall i, j, \end{array}$$

The Budget Adjust Mechanism

$$\begin{array}{rl} \max & \sum_{i \in B} (w_i + \delta_i) \log(\mathbf{u}_i^T \mathbf{x}_i) \\ \text{s.t.} & \sum_{i \in B} x_{ij} = \bar{s}_j, \quad \forall j \in G \\ & A_i \mathbf{x}_i \le \mathbf{b}_i, \forall i \in B, \\ & x_{ij} \ge 0, \quad \forall i, j, \end{array}$$

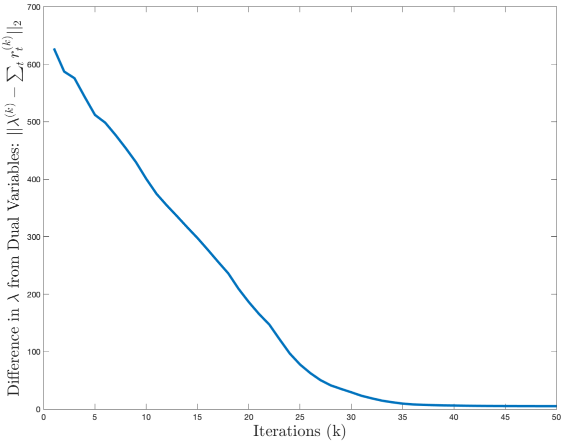
by adding $\delta_i \ge 0$'s at ideal levels, which itself becomes a fixed-point computation problem.

Theorem

There is one-one correspondence of an equilibrium price vector $\mathbf{p}^*$ *and a fixed-point solution of* δ_i^**s, that is,* $\delta_i^* = \mathbf{b}_i^T \mathbf{y}_i^*$, $\forall i$ *where* $\mathbf{y}_i^*$ *is the optimal multiplier vector of constraint* $A_i \mathbf{x}_i \le \mathbf{b}_i$ *with input* δ_i^**s.*

Implementation: There is no need to solve the problem to the individual level but a clusters of buyers who have similar behaviors/preferences.

Adjust $\delta_i^{k+1} = \mathbf{b}_i^T \mathbf{y}_i^k$ for all i iteratively where $\mathbf{y}_i^k$ the optimal multiplier vector of constraint $A_i \mathbf{x}_i \le \mathbf{b}_i$ with input δ_i^k.



Summary of the Model with Physical Constraints

- The equilibrium price vector exist under mild technical conditions.
- The problem seems able to be solved efficiently from a simple iterative procedure as described.

Summary of the Model with Physical Constraints

- The equilibrium price vector exist under mild technical conditions.
- The problem seems able to be solved efficiently from a simple iterative procedure as described.

Open questions

- Is the equilibrium price vector unique?
- Is the simple iterative procedure provably convergent?
- Is the fixed-point problem Tarski's type?
- Is the problem in the class of PPAD or there exists a polynomial time algorithm?