A Dynamic Near-Optimal Algorithm for Online Linear Programming

Yinyu Ye

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

Joint work with Shipra Agrawal and Zizhuo Wang

Information-Based Complexity and Stochastic Computation

September 17, 2014

Outline

- Online Linear Programming
- Main Results and Key Ideas
- Related and More Recent Work

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 sell 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 sell 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

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

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

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

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, $t = 1, \dots, n$ and

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

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

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

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.

Application Overview

- Revenue management problems: Airline tickets booking, hotel booking;
- Online network routing on an edge-capacitated network;
- Online combinatorial auction;
- Online adwords allocation

Model Assumptions

Main Assumptions

- $0 \leq a_{it} \leq 1$, for all (i, t);
- $\pi_t \geq 0$ for all t
- The data $(\mathbf{a}_t, \pi_t)$ arrive in a random order.

Model Assumptions

Main Assumptions

- $0 \leq a_{it} \leq 1$, for all (i, t);
- $\pi_t \geq 0$ for all t
- The data $(\mathbf{a}_t, \pi_t)$ arrive in a random order.

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 [Operations Research 2014]

Key Ideas: A Worst-Case Distribution Example

The proof of the negative result is based on a distribution of instances (the number of each types of columns is chosen according to certain distribution) with $m = 2^k$, and then show that no allocation rule can achieve $(1-\epsilon)$-optimality in expectation under randomized permutation.

Key Ideas: A Learning Algorithm is Needed

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

Key Ideas: A Learning Algorithm is Needed

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.

Key Ideas: A Learning Algorithm is Needed

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.

Key Ideas: A Learning Algorithm is Needed

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 is an "ideal price" vector:

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

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

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

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

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 $B \geq \frac{6m\log(n/\epsilon)}{\epsilon^3}$.

But this is one ϵ factor higher 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:

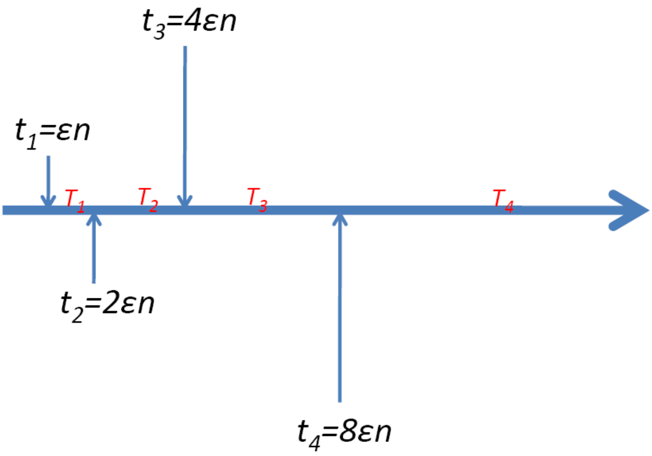
$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \leq (1 - h_\ell)\frac{\ell}{n} b_i & i = 1, \ldots, m \\ & 0 \leq x_t \leq 1 & t = 1, \ldots, \ell \end{array}$$

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 and 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 and 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 and Molinaro [2014]	$B \geq \frac{\log m}{\epsilon^2}$	Dynamic*

Table: Comparison of several existing results

Summary and Future Questions on OLP

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

Summary and Future Questions on OLP

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are near-optimal online algorithms for a very general class of online linear programs.

Summary and Future Questions on OLP

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are near-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 OLP

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are near-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 a factor.

Summary and Future Questions on OLP

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are near-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 a factor.
- Buy-and-sell or double market?

Summary and Future Questions on OLP

- $B = \frac{\log m}{\epsilon^2}$ is now a necessary and sufficient condition (differing by a constant factor).
- Thus, they are near-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 a factor.
- Buy-and-sell or double market?
- price-posting market?