Dual Interpretations and Duality Applications (continued I)

Yinyu Ye

Department of Management Science and Engineering

Stanford University

Stanford, CA 94305, U.S.A.

http://www.stanford.edu/˜yyye

(LY5th, Chapters 3.1-3.5, Wikipedia)

Two-Person Zero-Sum Game

Let P be the payoff matrix of a two-person, ”column” and ”row”, zero-sum game.

$$P = \begin{pmatrix} +3 & -1 & -4 \\ -3 & +1 & +4 \end{pmatrix}$$

Players usually use randomized strategies in such a game. A randomized strategy is a vector of probabilities, each associated with a particular decision.

Nash Equilibrium

In a Nash Equilibrium, if your (column) strategy is a pure strategy (one where you always play a single action), the expected payout for the (dominating) action that you are playing should be greater than or equal to the expected payout for any other action. If you are playing a randomized strategy, the expected payout for each action included in your strategy should be the same (if one were lower, you won't want to ever choose that action) and these payouts should be greater than or equal to the actions that aren't part of your strategy.

LP formulation of Nash Equilibrium

"Column" strategy:

$$\begin{array}{ll} \max & v \\ s.t. & v\mathbf{e} \le P\mathbf{x} \\ & \mathbf{e}^T\mathbf{x} = 1 \\ & \mathbf{x} \ge \mathbf{0}. \end{array}$$

"Row" strategy:

$$\begin{array}{ll} \min & u \\ s.t. & u\mathbf{e} \ge P^T\mathbf{y} \\ & \mathbf{e}^T\mathbf{y} = 1 \\ & \mathbf{y} \ge \mathbf{0}. \end{array}$$

They are dual to each other.

Multi-Firm LP Alliance I

Consider a finite set I of firms each of whom has operations that have representations as linear programs. Suppose the linear program representing the operations of firm i in I entails choosing an n-column vector $\mathbf{x} \geq \mathbf{0}$ of activity levels that maximize the firm's profit

$$\mathbf{c}^T\mathbf{x}$$

subject to the constraint that its consumption $A\mathbf{x}$ of resources minorizes its available resource vector $\mathbf{b}^i$, that is,

$$A\mathbf{x} \leq \mathbf{b}^i.$$

Multi-Firm LP Alliance II

An alliance is a subset of the firms. If an alliance S pools its resource vectors, the linear program that S faces is that of choosing an n-column vector $\mathbf{x} \geq \mathbf{0}$ that maximizes the profit $\mathbf{c}^T\mathbf{x}$ that S earns subject to its resource constraint

$$A\mathbf{x} \leq \mathbf{b}^S = \sum_{i \in S} \mathbf{b}^i.$$

Let V^S be the resulting maximum profit of S. The grand alliance is the set I of all firms.

$$\begin{array}{rlll} V^S := & \max & \mathbf{c}^T\mathbf{x} & \\ & \text{s.t.} & A\mathbf{x} & \leq \sum_{i \in S} \mathbf{b}^i, \\ & & \mathbf{x} & \geq \mathbf{0}, \end{array}$$

Multi-Firm LP Alliance III: Core

Core is the set of payment vector $\mathbf{z} = (z_1, \ ..., \ z_{|I|})$ to each company such that

$$\sum_{i \in I} z_i = V^I$$

and

$$\sum_{i \in S} z_i \geq V^S, \ \forall S \subset I.$$

Theorem 1 *For each optimal dual price vector for the linear program of the grand alliance, allocating each firm the value of its resource vector at those prices yields a profit allocation vector in the core.*

Robust Optimization I

Consider a linear program

$$\begin{array}{rl} \text{minimize} & (\mathbf{c}+C\mathbf{u})^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} \;=\; \mathbf{b}, \\ & \mathbf{x} \;\geq\; \mathbf{0}, \end{array}$$

where $\mathbf{u} \geq \mathbf{0}$ and $\mathbf{u} \leq \mathbf{e}$ is a state of Nature and beyond decision maker's control.

Robust Model:

$$\begin{array}{rl} \text{minimize} & \max_{\{\mathbf{u}\geq\mathbf{0},\ \mathbf{u}\leq\mathbf{e}\}}(\mathbf{c}+C\mathbf{u})^T\mathbf{x} \\ \text{subject to} & A\mathbf{x} \;=\; \mathbf{b}, \\ & \mathbf{x} \;\geq\; \mathbf{0}. \end{array}$$

Robust Optimization II

Nature's (primal) problem:

$$\begin{array}{rrcl} \text{maximize}_{\mathbf{u}} & \multicolumn{3}{l}{\mathbf{c}^T\mathbf{x} + \mathbf{x}^T C\mathbf{u}} \\ \text{subject to} & \mathbf{u} & \leq & \mathbf{e}, \\ & \mathbf{u} & \geq & \mathbf{0}. \end{array}$$

Dual of Nature's problem:

$$\begin{array}{rrcl} \text{minimize}_{\mathbf{y}} & \multicolumn{3}{l}{\mathbf{c}^T\mathbf{x} + \mathbf{e}^T\mathbf{y}} \\ \text{subject to} & \mathbf{y} & \geq & C^T\mathbf{x}, \\ & \mathbf{y} & \geq & \mathbf{0}. \end{array}$$

Robust Optimization III

Decision Maker's Robust Model:

$$\begin{aligned} \text{minimize}_{\mathbf{x},\mathbf{y}} \quad & \mathbf{c}^T\mathbf{x} + \mathbf{e}^T\mathbf{y} \\ \text{subject to} \quad & \mathbf{y} \;\geq\; C^T\mathbf{x}, \\ & A\mathbf{x} \;=\; \mathbf{b}, \\ & \mathbf{x},\ \mathbf{y} \;\geq\; \mathbf{0}. \end{aligned}$$

Distributionally Robust Optimization (DRO)

Consider a stochastic optimization Problem

$$\begin{array}{rrcl} \text{minimize} & E[f(\mathbf{x},\xi)] & & \\ \text{subject to} & A\mathbf{x} & = & \mathbf{b}, \\ & \mathbf{x} & \geq & \mathbf{0}, \end{array}$$

where ξ is a random vector from a distribution Ξ. Note that the expectation is a linear function of distributions, but in practice, Ξ is unknown to the decision maker.

DRO Model:

$$\begin{array}{rrcl} \text{minimize} & \max_{\Xi\in\mathcal{D}} E[f(\mathbf{x},\xi)] & & \\ \text{subject to} & A\mathbf{x} & = & \mathbf{b}, \\ & \mathbf{x} & \geq & \mathbf{0}. \end{array}$$

where $\mathcal{D}$ is the distribution uncertainty/ambiguity set constructed from the statistical moments or/and sample distributions, and it is a convex set of distributions.

A Online Resource Allocation Linear Programming Example

	order 1($t=1$)	order 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
Jacket	0	0	...	200
Socks	1	1	...	1000

Online Decision Making: Resource Allocation Linear Programming

In real applications, data/information is revealed sequentially, and one has to make decisions sequentially based on what is known – cannot wait for solving the offline problem. Consider problem:

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

Each bid/activity t requests a bundle of m resources, and the payment is π_t.

Online Decision Making: we only know $\mathbf{b}$ at the start, but

- the constraint matrix is revealed column by column 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.

Sequential Convex Programming Mechanism

$$
\begin{array}{lll}
\text{(SCPM):} & \text{maximize}_{x_t,\mathbf{s}} & \pi_t x_t + u(\mathbf{s}) \\
 & \text{s.t.} & \mathbf{a}_t x_t + \mathbf{s} = \mathbf{b} - \sum_{j=1}^{t-1} \mathbf{a}_j \bar{x}_j, \\
 & & 0 \le x_t \le 1, \\
 & & \mathbf{s} \ge \mathbf{0}.
\end{array}
$$

$\sum_{j=1}^{t-1} \mathbf{a}_j \bar{x}_j$: allocated resource vector before the new arrival.

Possible Concave Value Functions:

- Exponential: $u(s_i) = b \cdot (1 - \exp(-s_i/b)),$ for some positive constant b.
- Logarithmic: $u(s_i) = b \cdot \log(s_i),$ for some positive constant b.
- Quadratic:

$$
u(s_i) = \begin{cases} b \cdot (1 - (1 - s_i/b)^2) & 0 \le s_i \le b \\ b & s_i \ge b \end{cases} \quad \text{for some positive constant } b.
$$

Online Linear Programming

Main Assumptions

- The columns $\mathbf{a}_t$ arrive in a random order.
- We know the total number of columns n a priori.

Other technical assumptions

- $0 \leq a_{it} \leq 1$, for all (i, t);
- $\pi_t \geq 0$ for all t

The algorithm/mechanism quality is evaluated on the expected performance over all the permutations comparing to the offline optimal solution, i.e., an algorithm $\mathcal{A}$ is c-competitive if and only if

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

Comments on the Online Model

- The online approach is distribution-free. It allows for great robustness in practical problems. If the columns or arrivals are drawn $i.i.d.$ from a certain distribution (either known or unknown to the decision maker), then the first assumption is automatically met.
- The second assumption is necessary for one to obtain a near optimal solution. However, it can be relaxed to an approximate knowledge of n or the length of decision horizon.
- Both assumptions are reasonable and standard in many operations research and computer science applications.

Main Theorems

Theorem 2 *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 3 *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]

Comments on the Main Theorems

- The condition of B to hold the main result is independent of the size of $OPT(A, \pi)$ or the objective coefficients, and is also independent of any possible distribution of input data. Therefore, it's checkable.
- The condition on sample size $1/\epsilon^2$ is necessary as it is common in many learning-based algorithm.
- The condition is proportional only to $\log(n)$ so that it is way below to satisfy everyone's demand.

Key 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 $1 - \frac{\sqrt{\log(m)}}{\sqrt{B}}$.

Dual of the RA Problem

$$\begin{array}{ll} \text{minimize}_{\mathbf{x}} & \mathbf{b}^T\mathbf{p} + \sum_{j=1}^{n} z_j \\ \text{subject to} & \mathbf{p}^T\mathbf{a}_t - \pi_t + z_t \geq 0 \quad \forall j = 1, ..., n \\ & (\mathbf{p}, \mathbf{z}) \geq \mathbf{0} \end{array}$$

Strict Complementarity/Optimality Conditions:

$$x_t = \begin{cases} 0 & \text{if } \pi_t < \mathbf{p}^T\mathbf{a}_t \\ 1 & \text{if } \pi_t > \mathbf{p}^T\mathbf{a}_t \\ (0\; 1) & \text{if } \pi_t = \mathbf{p}^T\mathbf{a}_t \end{cases}$$

$\mathbf{p}$ are itemized prices of Goods!

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

One-Time Learning Algorithm

We start with a simple learning policy

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

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

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

- Determine the future allocation x_t as:

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

  as long as $a_{it} x_t \le b_i - \sum_{j=1}^{t-1} a_{ij} x_j$ for all i; otherwise, set $x_t = 0$.

One-Time Learning Algorithm Result

Theorem 4 *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 $B \geq \frac{6m\log(n/\epsilon)}{\epsilon^3}$.

But this is one ϵ factor higher than the lower bound...

Dynamic Price Updating 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 $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price vector is the optimal dual solution to the following linear program:

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

where

$$h_\ell = \epsilon\sqrt{\frac{n}{\ell}};$$

and this price vector is used to determine the allocation for the next immediate period.

Dynamic Price Updating Algorithm

In the dynamic price updating algorithm, we update the price at time ϵn, $2\epsilon n$, $4\epsilon n$ ... At time $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price is the optimal dual solution to the following linear program:

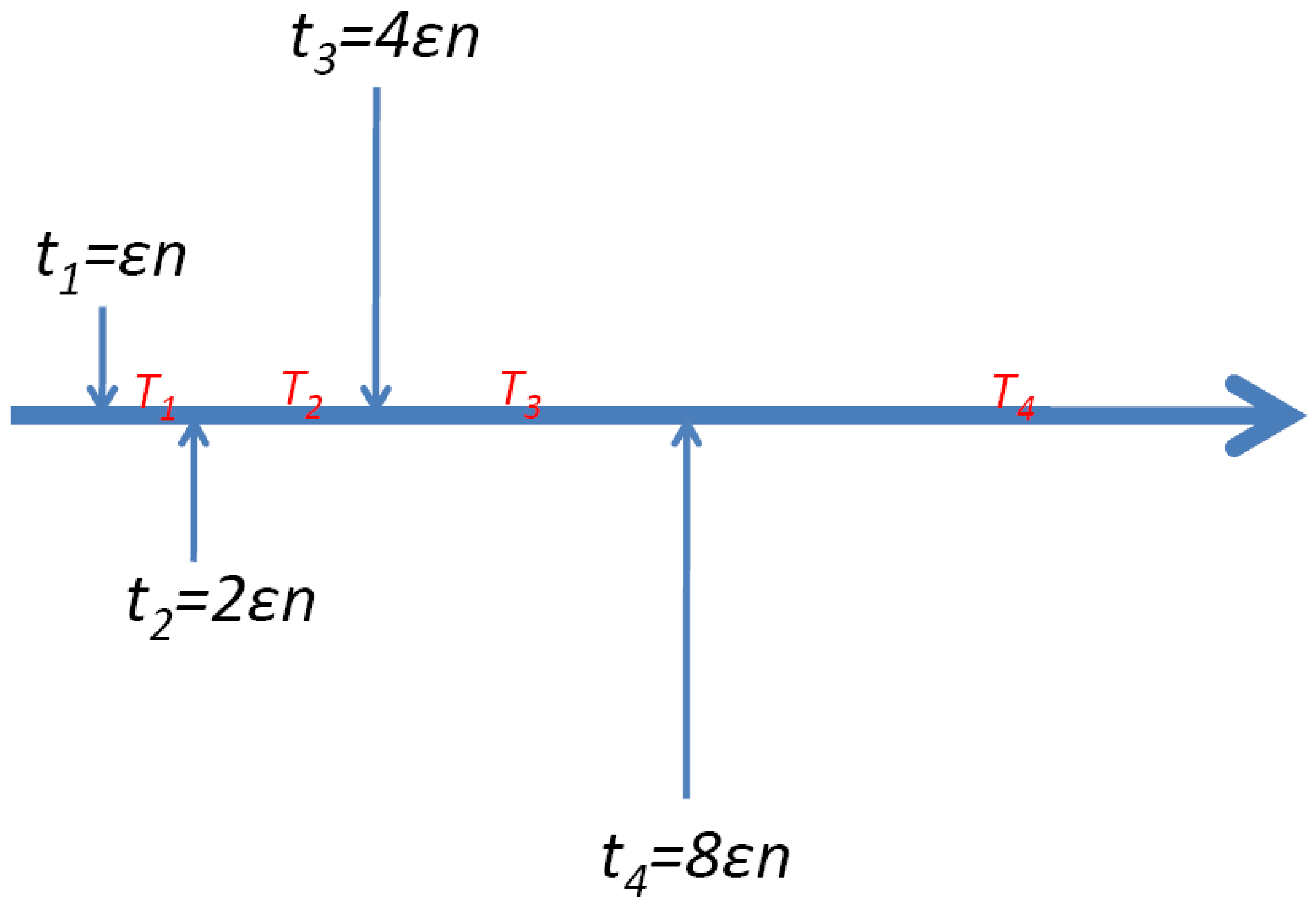
$$\begin{array}{lll} \text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\ \text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \le (1-h_\ell)\frac{\ell}{n} b_i & i = 1, ..., m \\ & 0 \le x_t \le 1 & t = 1, ..., \ell \end{array}$$

where

$$h_\ell = \epsilon\sqrt{\frac{n}{\ell}}$$

And this price 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.
- 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.

Dynamic/Adaptive Price Updating Algorithm

Similar to the dynamic price updating algorithm, we update the price at time ϵn, $2\epsilon n$, $3\epsilon n$ ... At time $\ell \in \{\epsilon n, 2\epsilon n, ...\}$, the price is the optimal dual solution to the following linear program:

$$
\begin{array}{lll}
\text{maximize}_{\mathbf{x}} & \sum_{t=1}^{\ell} \pi_t x_t & \\
\text{subject to} & \sum_{t=1}^{\ell} a_{it} x_t \le \frac{\ell}{n-\ell}\hat{b}_i & i = 1, ..., m \\
 & 0 \le x_t \le 1 & t = 1, ..., \ell
\end{array}
$$

where $\hat{b}_i$ is the remaining inventory at the beginning of the tth period. And this price is used to determine the allocation for the next immediate period.

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 1: 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).
- Thus, they are near-optimal online algorithms for a very general class of online linear programs under the permutation assumption.
- 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 worked under the iid inputs.
- Price-Posting multi-good model?