The Chinese University of HongKong, Shenzhen

Course Project: Online Linear Programming and Resource Allocation

DDA4300 Optimization in Data Science and Machine Learning

School of Data Science

Students:
Bingqi Zhang 120090365
Hao Ye 120090686
Haokun Zhao 120090255
Mengyang Lin 120090374
Qi Liu 120090027
Yu Zheng 120090683

Instructor
Prof. Yinyu Ye
TAs
Yushun Zhang
Chenhao Wang
Zongsen Yang

Click: Codesn

1 Problem Formulation

In this project, we consider the linear program for resource allocation of the form

$$\begin{aligned} \max_{\mathbf{x}\in\mathbb{R}^n} \quad & \boldsymbol{\pi}\cdot\mathbf{x} \\ \text{s.t.} \quad & A\mathbf{x}\leqslant\mathbf{b}, \\ & 0\leqslant\mathbf{x}\leqslant\mathbf{1}, \end{aligned} \tag{1}$$

where $A\in\mathbb{R}_+^{m\times n}, \pi\in\mathbb{R}_+^n$, and $\mathbf{b}\in\mathbb{R}_+^m$. The inputs of the online linear program are $\{\pi_j, \mathbf{a_j}\}$, where π_j represents j-th bid price, $\mathbf{b}$ represents the initial inventory level of each type of goods, and $\mathbf{a}_i$ of matrix A represents the amount of each type of goods requested by the i th bidder.

In the online setting, the decision-maker makes decisions on whether to accept a coming order without knowing the future information, and the decisions of allocation are irrevocable. Some assumptions on our data are as follows.

- all the samples $\{\pi_j, \mathbf{a_j}\}$ are randomly generated under the i.i.d. condition.
- $b_i = 1000$, for $i = 1, 2\cdots, 10$.
- $a_{ij}\in\{0,1\}$, for $i = 1, 2\cdots, 10$ and $j = 1, 2, \cdots, 10000$.
- $\bar{\mathbf{p}}$ is the fixed ground truth vector with $\bar{\mathbf{p}}_i = i$ for $i = 1, \ldots, 10$.
- $\pi_j = \bar{\mathbf{p}}^\top\mathbf{a_j} + randn(0, 0.2)$, where $randn(0, 0.2)$ represents the Gauss random variable with zero mean and standard deviation 0.2.

2 Algorithms Description

2.1 Sequential Linear Pari-Mutuel Mechanism (SLPM)

2.1.1 One-time Learning Algorithm

We consider the following partial linear problem that we wait for the first k bidders to arrive:

$$\begin{aligned} \max_{x_1,\cdots,x_k} \quad & \sum_{j=1}^{n}\pi_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{k} a_{ij}x_j \leq \tfrac{k}{n}b_i, \forall i = 1, 2, \cdots, m, \\ & 0\leq x_j\leq 1, \forall j = 1, \cdots, k. \end{aligned} \tag{2}$$

And then we use the dual prices (dual solutions) $\overline{\mathbf{y}}^k$ of the partial problem for future online decisions. For the first k customers, $x_j = 0$ for $j = 1, \ldots, k$. For future customers, the decision rule for x_j is:

$$x_j = 1, \text{if } \pi_j > \mathbf{a}_j^T\overline{\mathbf{y}}^k \text{ and sufficient inventory; otherwise } x_j = 0$$

2.1.2 Dynamic Learning Algorithm

Instead of computing the price only once, the dynamic learning algorithm will update the price every time the history doubles. The details are shown in **Algorithms 1**

2.2 Sequential Convex Pari-Mutuel Mechanism (SCPM)

The main difference between SLPM and SCPM is that SCPM considers the value of inventory remains for allocation. SCPM adds slack variable s to represent the remaining inventory and then adds a utility term $u(s)$ to measure its potential value.[1] [2]

$$\begin{aligned} \max_{x_1,\cdots,x_k,\mathbf{s}} \quad & \sum_{j=1}^{k}\pi_j x_j + u(\mathbf{s}) \\ \text{s.t.} \quad & \textstyle\sum_{j=1}^{k} a_{ij}x_j + s_i \leq \tfrac{k}{n}b_i,\ \forall\ i = 1, 2, \cdots, m \\ & 0\leq x_j\leq 1,\ \forall\ j = 1, 2, \cdots, k \\ & s_i\geq 0,\ \forall\ i = 1, 2, \cdots, m \end{aligned} \tag{3}$$

The dynamic learning algorithms for SLPM and SCPM are similar, the only difference is the programming of SCPM contains remaining resources s and its utility function $u(s)$. The details are shown in **Algorithm 2**

2.3 Action-History-Dependent Learning (AHDL)

In SLPM and SCPM, the dual price $\boldsymbol{p_t}$ depends on past inputs $\{(\pi_j, \boldsymbol{a_j})\}_{j=1}^{t}$, and the information of past decisions $(x_1, \cdots, x_j)$ is not considered. In contrast, AHDL integrates all the past decisions by stepwise updating the remaining resources[3]. For each coming order, we first decide whether to accept it based on the dual price derived from the last step and then update the resources if it is accepted. Finally, we update the dual price involving the information of the inputs of this coming order. The details are shown in **Algorithm 3**

2.4 Stochastic Gradient Descent Update

It is costly to use AHDL to update dual prices at each time step. The next algorithm addresses this problem. We first derive the dual problem of OPT.

$$\begin{aligned} \min_{\bar{\mathbf{y}}} \quad & \mathbf{b}^T\bar{\mathbf{y}} + \Sigma_{i=1}^{n}(\pi_i - \mathbf{a}_i^T\bar{\mathbf{y}})^+ \\ \text{s.t.} \quad & \bar{\mathbf{y}} \geq 0 \end{aligned}$$

Rather than solving this nonlinear problem, we solve another convex optimization problem (convexity and the connection between these problems are shown in the appendix). Here $\mathbf{d} = \mathbf{b}/n$

$$\begin{aligned} \min_{\bar{\mathbf{y}}} \quad & f(\bar{\mathbf{y}}) := \mathbf{d}^T\bar{\mathbf{y}} + \mathbb{E}(\pi - \mathbf{a}^T\bar{\mathbf{y}})^+ \\ \text{s.t.} \quad & \bar{\mathbf{y}} \geq 0 \end{aligned} \tag{4}$$

Under certain mild conditions, the objective f($\bar{\mathbf{y}}$) is differentiable with gradient

$$\nabla f(\bar{\mathbf{y}}) = \mathbb{E}(\mathbf{d} - \mathbf{a}\mathbb{I}_{\pi > \mathbf{a}^T\bar{\mathbf{y}}})$$

At each time step k, we approximate the gradient with $(\mathbf{d} - \mathbf{a_k}\mathbb{I}_{\pi_\mathbf{k} > \mathbf{a_k^T}\bar{\mathbf{y}}})$ and perform gradient decent method with step size $\frac{1}{\sqrt{k}}$. The details of the SGD algorithm are shown in **Algorithm 4**

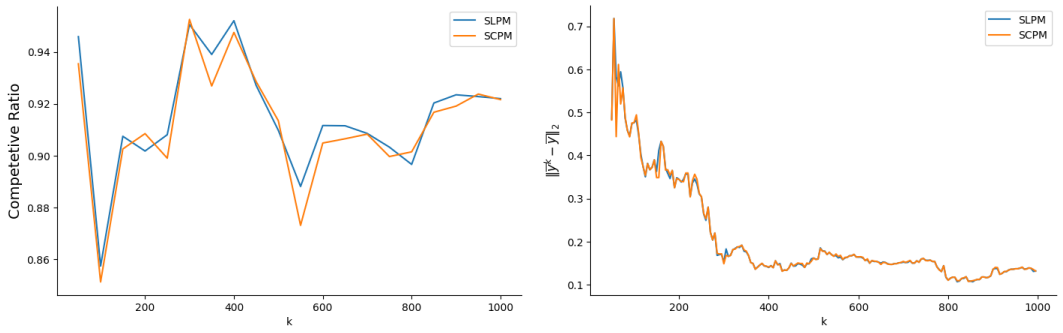
3 Experiments & Analysis

In this section, we use generated data to run experiments on different algorithms we discuss above and compare their performance and variants.

3.1 Comparison of SLPM and SCPM

3.1.1 One-time Learning Algorithm with different k

We run OLA(One-time Learning Algorithm) for SLPM and SCPM with different k, and compare their approximating ratios $(\frac{\pi x}{\pi x^*})$ and the convergence of dual prices.



(a) Competative Ratio

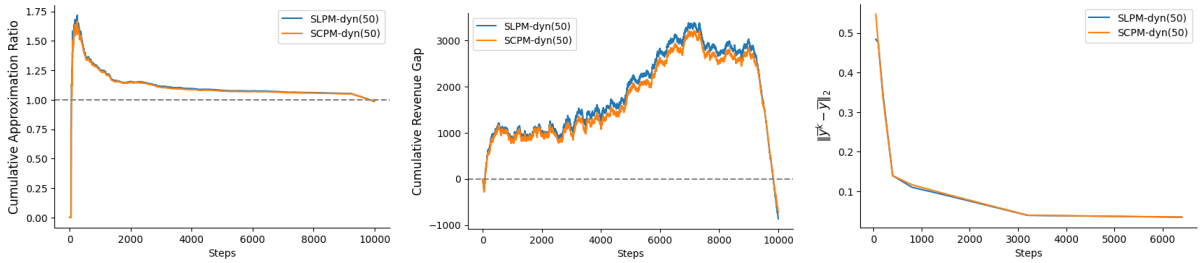
(b) Difference between online dual price and ground truth price

Figure 1: Comparison for One-time SLPM and SCPM

Figure 1b shows that the dual price will converge for both algorithms as k grows, but as shown in Figure 1a, the performance would not improve as k grows. This is because if we set k too large, we may lose many bids with high prices. If the k is too small, we can not have a good estimation of the dual price.

3.1.2 Comparison of DLA for SLPM and SCPM

We run DLA(Dynamic Learning Algorithm) for SLPM and SCPM. From Figure 2a and Figure 2b, the close competitive ratios and revenue gaps indicate these two algorithms have close performance under the i.i.d. condition. Besides, there is an interesting finding, SLPM allocates resources more aggressively, while SCPM is inclined to save resources for future allocations. Figure 2c shows the convergence of dual prices, the dual prices learned in these two algorithms are close. In summary, the performance of SLPM and SCPM is not much different under the i.i.d. assumption.

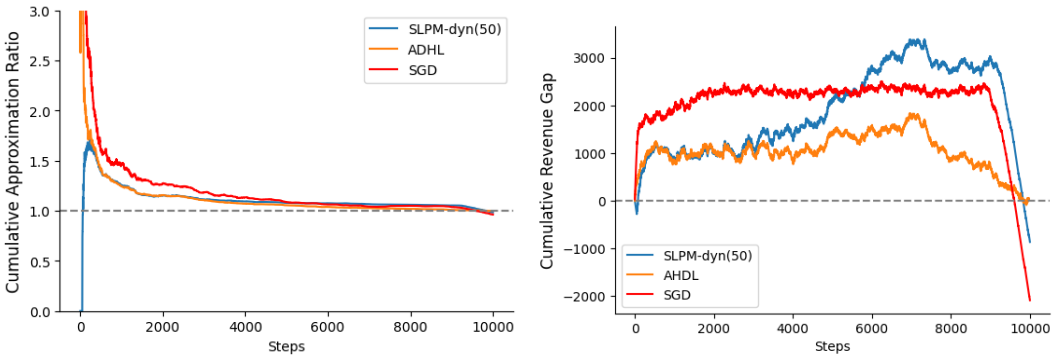


(a) Competitive Ratio of SLPM and SCPM (b) Cumulative Revenue Gap of SLPM and SCPM (c) Dual Price Convergence of SLPM and SCPM

Figure 2: Comparison of DLA for SLPM and SCPM

3.2 Comparison of DLA-SLPM, AHDL and SGD

We run DLA-SLPM, AHDL and SDG to compare their performance.



(a) Competitive Ratio of SLPM, AHDL and SGD (b) Cumulative Revenue Gap of SLPM, AHDL and SGD

Figure 3: Comparison of DLA-SLPM, AHDL and SGD

According to Figure 3, AHDL has the best performance among these three algorithms, achieving a competitive ratio of 99.4%, this is because AHDL utilizes the accurate information of inventory level and updates the dual price stepwise, while SLPM only approximates the inventory level by normalising the entire resources. And we also recognize that although AHDL performs well, it is time-consuming since it needs to solve a large-scale LP to update the dual price at each step. Compared to AHDL, SGD is fast because it requires less computation when updating the dual prices.

4 Coping Strategy when i.i.d. Assumption is Violated

A key condition for the above algorithms to work is i.i.d. condition on inputs $\{(\pi_j, \boldsymbol{a_j})\}$. Experiments show that algorithms perform poorly under the non-i.i.d. condition. We proposed an algorithm called **Window Shift Learning** algorithm to cope with the non-i.i,d. condition.

4.1 Window Shift Learning Algorithm (WSL)

Like AHDL, WSL also updates the remaining resources after each decision is made. The key difference between AHDL and WSL is that WSL focuses on only the near-historical data when updating the dual price. The intuition behind this algorithm is that when the i.i.d. assumption for bids doesn't hold, the most recent bids can provide

Algorithm 5 Window Shift Learning Algorithm

Initialization: $k, \theta, \gamma, \boldsymbol{p_0} \leftarrow \infty$,
for $w \in 0, 1, \cdots, n/k$ **do**
  $\theta \leftarrow TC(w, \gamma)$
  for $j \in wk, wk+1, \cdots, wk+k-1$ **do**
    if $\pi_j \geq a_j p_{w-1}$ and sufficient inventory **then**
      $x_j = 1$
    else
      $x_j = 0$
    end if
    Update the remaining resources
      $b_i^{(t)} = b_i^{(t-1)} - a_j x_j \quad \text{for} \quad i = 1, 2, \cdots, m$
  end for
  Update the dual price p_{w+1} by solving the dual problem of the following linear programming

$$\begin{aligned} \max \quad & \sum_{j=wk}^{wk+k-1} \pi_j x_j \\ \text{s.t.} \quad & \sum_{j=w}^{w+k-1} a_{ij} x_j \leq \frac{\theta k}{n - wk} b_i^w \quad \forall i \in 1, 2, \cdots, m \\ & 0 \leq x_j \leq 1 \quad \forall j = wk, wk+1, \cdots, wk+k-1 \end{aligned} \tag{5}$$

end for=0

Algorithms	Competitive Ratio			Average Running Time
	Increasing	Decreasing	Fluctuating	
WSL	**0.864**	**0.960**	**0.875**	37.5s
OLA-SLPM	0.467	0.035	0.775	10.2s
OLA-SCPM	0.451	0.035	0.775	10.7s
DLA-SLPM	0.452	0.028	0.787	29.8s
DLA-SCPM	0.451	0.028	0.788	30.1s
AHDL	0.786	0.732	0.834	12min
SGD	0.463	0.798	0.670	3.1s

Table 1: Comparison of different algorithms on dependent datasets

more valuable information about current decisions. Our policy is to estimate the dual price using only the most recent inputs $\{\pi_j, \boldsymbol{a_j}\}_{j=t-k-1}^{t-1}$ within a moving window, forgetting all the data outside the window. Moreover, under the non-i.i.d condition (e.g $\{\pi_j, \boldsymbol{a_j}\}$ is a sequence of random work), the decision maker would benefit if the variation trend of the bids is known. Aiming to capture the trend of bids, we propose a method called Trend Capture. The intuition behind this method is that if the bid prices are increasing, we would shrink the resources using the shrinkage factor θ to overestimate the dual price so that it would be more inclined to reject an order. On the contrary, if the decreasing trend of bids is detected, we would loose the resources to lower the dual price so as to accept more orders in the near future. There would come a myopia issue if we only estimate the price in the near future. We apply a trick to cope with this issue, if the price is increasing and we still have lots of remaining resources, we would be very optimistic about the far future payoff, thus we should strengthen the shrinkage of resources to overestimate a higher dual price and vice versa. The details of this algorithm are shown in **Algorithm 5** and **6**.

4.2 Experiments on Dependent Dataset

We run different algorithms on three non-i.i.d. datasets (bid prices are increasing, decreasing and fluctuating respectively). Table 1 shows that algorithms except WSL and AHDL perform poorly on increasing and decreasing datasets. Although AHDL has a relatively higher performance, it is time-consuming. Among all algorithms, WSL has the best performance and relatively lower computation. We also recognize that compared to other algorithms, WSL algorithm can capture the trend of bids because of two reasons. The first one is WSL uses the most recent information (within the window) instead of entire historical data which may be outdated. The second one is it intentionally overestimates or underestimates the dual price under different circumstances by restricting the resources within the window.

5 Acknowledgement

We would like to thank Prof. Ye for the amazing DDA4300 and fruitful discussions with us on this project. We would also like to thank TAs for their generous help. We are glad to apply the knowledge we learnt in the class to tackle a real-world problem via this project.

References

[1] S. Agrawal, E. Delage, M. Peters, Z. Wang, and Y. Ye, "A unified framework for dynamic prediction market design," *Operations research*, vol. 59, no. 3, pp. 550–568, 2011.

[2] M. Peters, A. M.-C. So, and Y. Ye, "Pari-mutuel markets: Mechanisms and performance," in *Internet and Network Economics: Third International Workshop, WINE 2007, San Diego, CA, USA, December 12-14, 2007. Proceedings 3*. Springer, 2007, pp. 82–95.

[3] X. Li and Y. Ye, "Online linear programming: Dual convergence, new algorithms, and regret bounds," *Operations Research*, vol. 70, no. 5, pp. 2948–2966, 2022.

6 Appendix

6.1 Pseudocode of Algorithms

Algorithm 1 Dynamic Learning - SLPM

SET $x_1, \cdots, x_{50} = 0$; dual price $p = 0$
for $k = 50, 100, 200, \cdots$ **do**
  solve SLPM with k decision variables $x_1, \cdots, x_k$
  compute dual price $\hat{\mathbf{y}}^k$
  make allocation via: $x_j = 1$, if $\pi_j > \mathbf{a}_j^T \hat{\mathbf{y}}^k$; $x_j = 0$, otherwise
end for
OUTPUT $x_1, \cdots, x_n$ =0

Algorithm 2 Dynamic Learning - SCPM

SET $x_1, \cdots, x_{50} = 0$; dual price $p = 0$
for $k = 50, 100, 200, \cdots$ **do**
  solve SCPM with k decision variables $x_1, \cdots, x_k$
  compute dual price $\hat{\mathbf{y}}^k$
  make allocation via: $x_j = 1$, if $\pi_j > \mathbf{a}_j^T \hat{\mathbf{y}}^k$; $x_j = 0$, otherwise
end for
OUTPUT $x_1, \cdots, x_n$ =0

Algorithm 3 Action-History-Dependent Learning

Initialize $b^{(0)} = b; t = 0, \boldsymbol{p_0} = \infty$;
for $t = 0, 1, 2, \cdots, n$ **do**
  if $\pi_t \geq a_t^\top p_t$ **then**
    $x_t = 1$
  else
    $x_t = 0$
  end if
  Update the remaining resources
    $b_i^{(t)} = b_i^{(t-1)} - a_j x_j \quad \text{for} \quad i = 1, 2, \cdots, m$
  Update the dual price p_{t+1} by solving the dual problem of the following linear programming

$$\begin{aligned} &\max \sum_{j=1}^{t} \pi_j x_j \\ &\text{s.t.} \ \sum_{j=1}^{t} a_{ij} x_j \leq \frac{t b_i^{(t)}}{n-t}, \quad i = 1, \ldots, m \\ &\qquad 0 \leq x_j \leq 1, \quad j = 1, \ldots, t \end{aligned}$$

end for=0

Algorithm 4 Stochastic Gradient Descent for online allocation problem

$\bar{\mathbf{y}}^{(0)} \leftarrow 0$
$x_1 \leftarrow 0$
for $k = 1$ to $n-1$ **do**
  $\nabla f(\bar{\mathbf{y}}^{(k)}) \leftarrow \mathbf{d} - \mathbf{a}_k \mathbb{I}_{\pi_k > \mathbf{a}_k^T \bar{\mathbf{y}}^{(k)}}$
  $\bar{\mathbf{y}}^{(k)} \leftarrow \bar{\mathbf{y}}^{(k-1)} - \frac{1}{\sqrt{k}} \nabla f(\bar{\mathbf{y}}^{(k)})$
  if $\pi_{k+1} > \mathbf{a}_{k+1}^T \bar{\mathbf{y}}^{(k)}$ and remaining goods > 0 **then**
    $x_{k+1} \leftarrow 1$
    Update remaining inventory with $\mathbf{a}_k$
  else
    $x_{k+1} = 0$
  end if
end for=0

Algorithm 6 Trend Capture

Input: w, γ
Initialize $\delta = 0$
for $t \in 1, \cdots, w$ **do**
  $\delta = \delta + \frac{p_t - p_{t-1}}{P_{t-1}} \gamma^{w-t}$
end for
if $\delta > 0$ **then**
  $\theta \leftarrow (1-\delta)\frac{b-b^w}{b}$
else
  $\theta \leftarrow (1+\delta)\frac{b+b^w}{b}$
end if=0

6.2 Proof of 2.2

Let the Lagrangian function be $L(x, s, \mu, \eta) = \pi^T x + u(s) + \mu^T(e - x) + y^T(b - Ax - s) + \lambda^T x + \eta^T s$, where $A = (a_{ij})_{m \times n}$
The first-order KKT conditions for optimality are as follows:

$$\begin{aligned} \nabla_x L = \pi - \mu - A^T y + \lambda = 0,\ \nabla_s L = \nabla u(s) - y + \eta = 0 \\ Ax + s = b,\ 0 \le x \le 1,\ s \ge 0 \\ \mu \ge 0 \\ \mu_i(1 - x_i) = 0,\ \forall i = 1, 2, \ldots, n \\ \lambda_i x_i = 0,\ \forall i = 1, 2, \ldots, n \\ \eta_j s_j = 0,\ \forall j = 1, 2, \ldots, m \end{aligned}$$

Proof: Indeed, $u(s)$ can be represented as $h(x) = u(b - Ax)$. Since $b - Ax$ is concave and u is strictly concave, $h(x)$ is strictly concave, that is the objective function is strictly concave.

Now we know this problem is a convex optimization problem since the objective function is strictly concave and the feasible set is a convex set. Thus, the first-order KKT conditions are sufficient for global optimality. Furthermore, these constraints satisfy Slater conditions since constraints have no nonlinear conditions, and then we know these KKT conditions are necessary for the global optimality of this program. Therefore, solutions to KKT conditions are equivalent to the optimal solutions to this problem.

Let s^* be an optimal solution and y^*, η^* be optimal multipliers. According to the above KKT conditions, since $(\nabla u(s))_i$ is sufficiently large at $s_i = 0, s_i$ can not reach 0, which means $\eta_i^* = 0\ \forall i$. Therefore, $y^* = \nabla u(s^*)$. Thus, if s^* is unique, then y^* is unique. Next, we show the primal optimal solution s^* is unique.

As mentioned above, the objective function is a continuous function with respect to x. The constraints on x imply that x is bounded and closed. Thus, the objective function has its maximum. Meanwhile, this is a convex optimization problem. Therefore, the optimal solution s^* is unique, i.e. y^* is unique.

Hence, we proved the optimal prices are unique. □

6.3 Proof of 2.4

Proof:

We first derive the dual problem of (1),

$$\begin{aligned} minimize_{\bar{\mathbf{y}}} \quad & \mathbf{b}^T\bar{\mathbf{y}} + \Sigma_{i=1}^n t_i \\ s.t. \quad & \bar{\mathbf{y}} \geq 0, \\ & \mathbf{t} \geq 0, \\ & \mathbf{t} \geq \pi - \mathbf{a}^T\bar{\mathbf{y}} \end{aligned}$$

which is equivalent to

$$\begin{aligned} & minimize_{\bar{\mathbf{y}}} \ \mathbf{b}^T\bar{\mathbf{y}} + \Sigma_{i=1}^n(\pi_i - \mathbf{a}_i^T\bar{\mathbf{y}})^+ \\ & s.t. \quad \bar{\mathbf{y}} \geq 0 \end{aligned}$$

Let $\mathbf{d} = \mathbf{b}/n$, then we have

$$\begin{aligned} & minimize_{\bar{\mathbf{y}}} \ f_n(\bar{\mathbf{y}}) := \mathbf{d}^T\bar{\mathbf{y}} + \frac{1}{n}\Sigma_{i=1}^n(\pi_i - \mathbf{a}_i^T\bar{\mathbf{y}})^+ \\ & s.t. \quad \bar{\mathbf{y}} \geq 0 \end{aligned}$$

Consider the following problem

$$\begin{aligned} & minimize_{\bar{\mathbf{y}}} \ f(\bar{\mathbf{y}}) := \mathbf{d}^T\bar{\mathbf{y}} + \mathbb{E}(\pi - \mathbf{a}^T\bar{\mathbf{y}})^+ \\ & s.t. \quad \bar{\mathbf{y}} \geq 0 \end{aligned} \tag{6}$$

$(\pi_j, \mathbf{a}_j)$ is a sequence of i.i.d. random vectors, then

$$\mathbb{E}[\mathbf{d}^T\bar{\mathbf{y}} + \tfrac{1}{n}\Sigma_{i=1}^n(\pi_i - \mathbf{a}_i^T\bar{\mathbf{y}})^+] = \mathbf{d}^T\bar{\mathbf{y}} + \mathbb{E}[(\pi - \mathbf{a}^T\bar{\mathbf{y}})^+]$$

Thus, $\mathbb{E}[f_n(y)] = f(y)$. f_n can be considered as a sample average approximation of f.
Note that $(\pi - \mathbf{a}^T\bar{\mathbf{y}})^+$ is convex, and since expectation preserves convexity, then $f(\bar{\mathbf{y}})$ is convex. The feasible set is a convex set. Hence, this is a convex optimization problem. □

6.4 Additional plots for dependent datasets

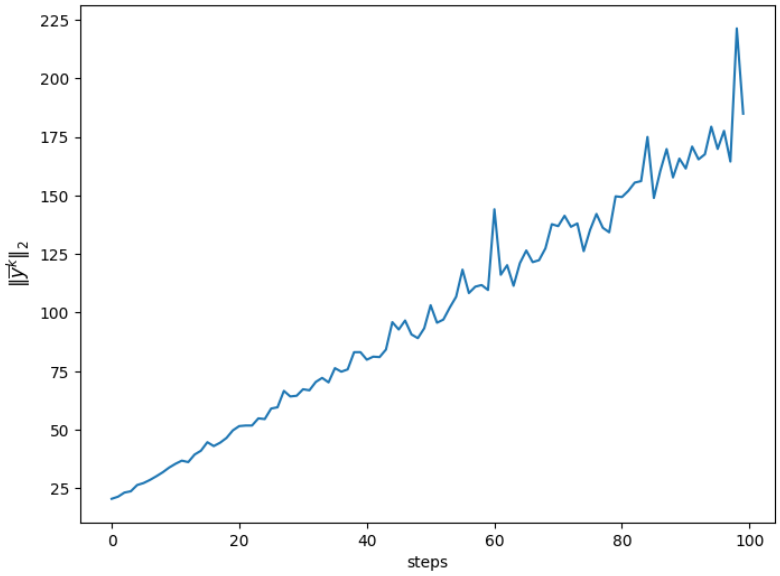


Figure 4: L2 norm of dual price on increasing dataset

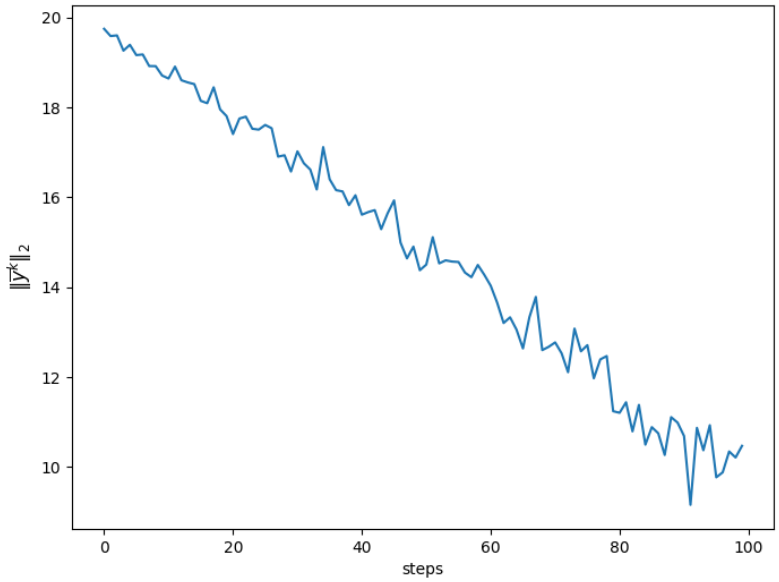


Figure 5: L2 norm of dual price on decreasing dataset

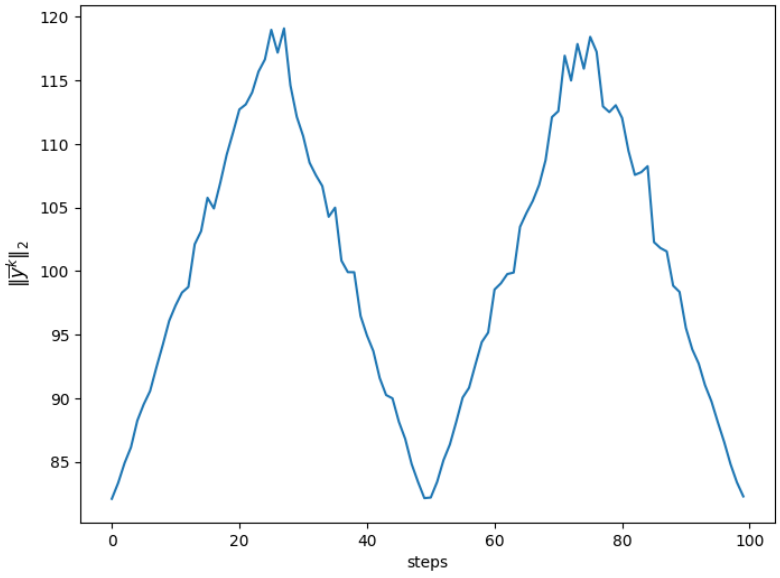


Figure 6: L2 norm of dual price on fluctuate dataset