Learning-Augmented Algorithms

Candidate Number: 16436

A Dissertation submitted to the Department of Mathematics of the London School of Economics and Political Science

April 1, 2022

Summary

In this dissertation we explore a recent line of research that studies a new type of algorithm called learning-augmented algorithms. These are online algorithms that include machine-learned predictions with the aim of improving their performance. We focus on the classical ski rental problem to see how leaning-augmented algorithms achieve this using the work developed by Kumar et al. (2018) and Gollapudi & Panigrahi (2019).

Contents

Summary ii

1 Introduction 1

2 Ski Rental without Prediction 4

2.1 Deterministic Solution: 4

2.2 Randomised Solution: 4

3 Ski Rental with Prediction 6

3.1 A Simple Consistent, Non-Robust Algorithm 7

3.2 A Deterministic Robust and Consistent Algorithm 7

3.3 A Randomised Robust and Consistent Algorithm 9

4 Ski Rental with Multiple Predictions 10

4.1 A Deterministic Algorithm with Two Predictions 11

4.2 A Randomised Algorithms with Two Predictions 12

5 Conclusion and Results 13

A Appendix 14

A.1 Proof of Theorem 2 14

A.2 Proof of Theorem 4 18

A.3 Proof of Theorem 5 19

Bibliography 22

Chapter 1

Introduction

A CPU scheduler receives a number of jobs at the same time has to decide which one to process without knowing their running time and the future jobs. These types of problems are addressed in a field of mathematics called optimisation under uncertainty and there are two computational paradigms that tackle this.

The first is machine learning, which deals with uncertainty by making predictions of the future using past data. The second is a certain type of algorithms called online algorithms (Borodin & El-Yaniv (1998)). These are algorithms that are designed to solve a problem without knowing the input in advance. Their performance is compared with that of their counterpart, offline algorithms, which do know the input prior. The aim is to optimise the decisions that the online algorithm makes such that its performance matches that of the offline algorithm. Following (Lykouris & Vassilvtiskii (2018); Kumar et al. (2018)) we measure the performance with a metric called the Competitive Ratio.

Definition 1 (Competitive Ratio)**.** The Competitive Ratio is the ratio of the cost of the online algorithm in the worst-case scenario and the optimal cost of the offline algorithm, over all possible inputs.

This ratio shows us how many times worse the online algorithm performs compared to the offline algorithm. The problem with online algorithms is that they assume the worst-case scenario over all possible inputs and this approach is too pessimistic for some real-world settings.

Therefore, recent research has decided to incorporated machine learning predictions into online algorithms (Medina & Vassilvitskii (2017); Wei & Zhang (2020); Anand et al. (2020); Kumar et al. (2018)) in order to improve their performance. These algorithms are called learning-augmented algorithms. They have to be *consistent* (if the predictions are accurate the performance should be similar to that of the offline algorithm) and *robust* (if the predictions are inaccurate the performance cannot be worse than that of the online algorithm without predictions). Learning-

augmented algorithms benefit from the ability that machine learning has to adapt to input data and the guarantee for correctness and performance of online algorithms.

This dissertation uses the classical ski rental problem (Karlin et al. (1994); Lotker et al. (2008); Khanafer et al. (2013); Kodialam (2014)) in order to show how learning-augmented algorithms work. In this problem a skier wants to ski for the length of the ski season, and he can either rent the skis for one day or buy them. He wishes to minimise the cost he spends on the skis. The problem is that he does not know in advance the length of the ski season, so an algorithm is designed that deals with this uncertainty and minimises the cost. Different variations of this problem have been studied such as dynamic TCP acknowledgement (Karlin et al. (2003)), buying parking permits (Meyerson 2005) and snoopy caching (Karlin et al. 1988).

In Chapter 2, we study the classical online algorithms for the ski rental problem. The first section analyses a deterministic 2-competitive algorithm, and the second section analyses a randomised e/(e-1)-competitive algorithm. In the third chapter, we combine a machine learning model with an online algorithm to produce a learning-augmented algorithm. In the first section, we start with a 1-consistent algorithm, then in the next section we manipulate the previous algorithm so it is robust. Finally, we improve the consistency and robustness bounds by converting it into a randomised robust and consistent algorithm. In Chapter 4, we introduce various machine learning models into our learning-augmented algorithm to increase the accuracy of the predictions. In Section 4.1 we assume one of the machine learning models has zero prediction error. In Section 4.2 we assume a more realistic scenario where all the machine learning models have a prediction error. Last, in Section 4.3 we make the algorithm with multiple machine learning models robust.

Chapters 2 and 3 burrow concepts, theorems and proofs from Kumar et al. (2018) and Chapter 4 burrows them from Gollapudi & Panigrahi (2019). The aim of this dissertation is to explain learning-augmented algorithms and does not include any original research.

Notation:

- x is the length of the ski season
- b is the cost of buying the skis
- 1 is the cost of renting the skis for one day
- ALG is the cost of the online or learning-augmented algorithm analysed.
- OPT is the solution obtained by the optimal offline algorithm for input x.
- Competitive Ratio $= \sup_x \frac{ALG}{OPT}$ ($\sup ALG$ *is the cost in the worst-case scenario*)

Assumption. *We assume that b is known and $b > 1$ (it is more expensive to buy than to rent).*

Suppose we know the length of the season is 10 days ($x = 10$) and the cost of buying the skis is 5 ($b = 5$), then the optimal solution for the skier would be to buy the skis from the first day. However, if the length of the season is 3 days ($x = 3$) then the optimal solution would be to rent the skis and incur only a cost of 3. Therefore, the optimum cost is $\min(b, x)$.

Chapter 2

Ski Rental without Prediction

First, we are going to examine online algorithms that do not take advantage of machine-learned predictions.

2.1 Deterministic Solution:

Let us start with this simple algorithm. In this algorithm, when b is for example 4 we are going to rent until the third day and on the fourth day we decide to buy the skis.

Algorithm 1 Deterministic algorithm

```
while current day is not equal to b do
    Rent skis
end while
Buy skis
```

Theorem 1. *Competitive Ratio of Algorithm 1 is 2*

Proof. If $x < b$, then OPT = ALG = x.
Else, if $x \geq b$, then OPT = b and ALG = $b - 1 + b = 2b - 1$ [1].
Hence, the Competitive Ratio = $\sup_x \frac{ALG}{OPT} \underset{\text{x}\geq\text{b}}{=} \frac{2b-1}{b} = 2$ □

2.2 Randomised Solution:

Notation:

- k is the day we decide to buy the skis

[1] The skier rents for $b - 1$ days and on the bth day he buys.

The worst-case for Algorithm 1, the deterministic algorithm, is when we decided to buy on day k and $x = k$, meaning we decided to buy on last day of the season. Now we are going to randomly choose the day we buy the skis. Our objective would be to make the expected cost of this algorithm lower than that of Algorithm 1.

Algorithm 2 Randomised algorithm

k given from randomised function
Rent for $k-1$ days
Buy on kth day

Theorem 2. *Competitive Ratio of Algorithm 2 is* $\frac{e}{e-1} \approx 1.58$

Proof. [2] Let p_k be the probability of buying on day k. We can solve this problem by modelling it as an infinite linear program. Our goal is to minimise the Competitive Ratio for all possible values of x. We carry this proof in five steps.

Step 1: We set up a linear program for a particular value of b.

Step 2: We simplify this linear program by removing any redundant inequalities.

Step 3: We prove that the c from our linear program is the lowest possible c.

Step 4: We solve the linear system we are left with.

Step 5: We generalise for b and solve the resulting linear system.

We now have the values of p_k, which we use to find c. □

[2] For full proof please visit the appendix

Chapter 3

Ski Rental with Prediction

A genie (machine learning model) is now introduced turning the online algorithm into a learning-augmented algorithm. This machine learning model is going to use past data such as the weather, the altitude of the mountains, etc and give us a prediction of the length of the ski season. However, the accuracy of the genie is unknown.
Notation:

- y = predicted number of days by the genie
- $\eta = |x - y|$ = prediction error

The algorithms discussed in this section must have three different properties:

- **Independence:** The algorithm should be independent of the machine-learned prediction and make no assumptions about the prediction's error types and distribution, i.e. the performance should not depend on η. Like this if we either improve the machine-learned prediction or the algorithm independently, we have an improvement in performance. Moreover, this allows the same algorithm to be used in different cases where the error types and distributions are different.
- **Consistency:** If the prediction is good, then the algorithm should perform close to the best offline algorithm. We say that an algorithm is β-consistent if when $\eta = 0$ the Competitive Ratio = β.
- **Robustness:** If the prediction is bad, then the algorithm should perform close to that of an algorithm that does not use predictors. We say that an algorithm is γ-robust if the Competitive Ratio is less than or equal to γ for all possible inputs. This means that the Competitive Ratio is bounded over all possible inputs.

From now on we are not going to analyse the Competitive Ratio of the algorithm but its consistency and robustness.

3.1 A Simple Consistent, Non-Robust Algorithm

In this algorithm, the genie is trusted and we regard his prediction as being 100% accurate. If the genie predicts that the ski season is going to be longer than the cost of buying the skis, we buy the skis on the first day. Otherwise, we rent until the end of the ski season.

Algorithm 3 Simple 1-consistent algorithm

if $y \geq b$ **then**
  Buy on the first day
else
  Keep renting for all the ski season
end if

Theorem 3. *Algorithm 3 is* 1*-consistent but not robust*

Proof. The proof is divided into the four possible cases of the prediction values y and the actual values x:

$$y \geq b, x \geq b \Rightarrow ALG = b = OPT, \quad \text{Competitive Ratio} = 1$$
$$y < b, x < b \Rightarrow ALG = x = OPT, \quad \text{Competitive Ratio} = 1$$
$$y \geq b, x < b \Rightarrow ALG = b \geq x + (y - x) = x + \eta = OPT + \eta,$$
$$\text{Competitive Ratio} = \frac{b}{x}$$
$$y < b, x \geq b \Rightarrow ALG = x < x + (b - y) = b + (x - y) = b + \eta = OPT + \eta,$$
$$\text{Competitive Ratio} = \frac{x}{b}$$

The Competitive Ratio $= 1$ when $y = x \quad (\eta = 0)$, hence it is 1-consistent. However, in the fourth case if $x >> b$, the algorithm is not robust as the Competitive Ratio $= \frac{x}{b}$ is not bounded. □

3.2 A Deterministic Robust and Consistent Algorithm

Now, in order to make the algorithm more robust the hyperparameter λ is introduced to denote how much the algorithm trusts the prediction model.
Let $\lambda \in (0, 1)$

- If λ is small (close to 0), we trust the genie.
- If λ is large (close to 1), we don't trust the genie.

This will allow the algorithm to trade-off between robustness and consistency.

Algorithm 4 Deterministic robust and consistent algorithm

if $y \geq b$ **then**
  Buy on day $\lceil \lambda b \rceil$
else
  Buy on day $\lceil \frac{b}{\lambda} \rceil$
end if

Theorem 4. *Algorithm 4 is* $(1+\lambda)$*-consistent and* $\left(\frac{1+\lambda}{\lambda}\right)$*-robust.*

Proof. [1] We first prove that the Competitive Ratio $\leq \frac{1+\lambda}{\lambda}$. We calculate the Competitive Ratio for the different cases of the prediction of the Genie and the length of the length of the season. We see that $\frac{1+\lambda}{\lambda}$ is an upper bound.

In order to prove that the Competitive Ratio $\leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$, we do the same steps as before but this time taking into account the prediction error in our upper bound.

Hence, the Competitive Ratio can at most be $\frac{1+\lambda}{\lambda}$ or $(1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$. The algorithm is $\frac{1+\lambda}{\lambda}$-robust as the Competitive Ratio will always be smaller than $\frac{1+\lambda}{\lambda}$ for all η. The algorithm is $(1+\lambda)$-consistent as the Competitive Ratio $= (1+\lambda)$ when $\eta = 0$. □

We can see that a lower λ trusts the genie more and gives us a higher consistency, whereas a higher λ trusts the genie less and gives us a higher robustness. This is called the consistency-robustness tradeoff, where an algorithm becomes more robust in the expense of becoming less consistent or vice versa.

[1] For full proof please visit the appendix

3.3 A Randomised Robust and Consistent Algorithm

In this algorithm we are going to use randomised algorithms to give us better consistency and robustness bounds.

Algorithm 5 Randomised robust and consistent algorithm

Let $\lambda \in (\frac{1}{b}, 1)$

if $y \geq b$ **then**

Let $k \leftarrow \lfloor \lambda b \rfloor$

Define $q_i \leftarrow (\frac{b-1}{b})^{k-i} \frac{1}{b(1-(1-1/b)^k}$ for all $1 \leq i \leq k$

Choose j $\in \{1, 2, ...,$k$\}$ *randomly from distribution defined by* q_i

Buy on day j

else

Let $l \leftarrow \lceil \frac{b}{\lambda} \rceil$

Define $r_i \leftarrow (\frac{b-1}{b})^{l-i} \frac{1}{b(1-(1-1/b)^l}$ for all $1 \leq i \leq l$

Choose j $\in \{1, 2, ...,$l$\}$ *randomly from distribution defined by* r_i

Buy on day j

end if

Theorem 5. *Algorithm 5 is* $\left(\frac{\lambda}{1-e^{-\lambda}}\right)$*-consistent and* $\left(\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}\right)$*-robust.*

Proof. [2] As before we are going to consider the different cases for y and x. In each case, we are going to calculate the expected cost of the algorithm and determine an upper bound. This will give us the upper bounds of the Competitive Ratio which will be $\left(\frac{\lambda}{1-e^{-\lambda}}\right)(1+\frac{\eta}{OPT})$ and $\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$. Hence, when $\eta = 0$ the Competitive Ratio $= \frac{\lambda}{1-e^{-\lambda}}$ so the algorithm is $\left(\frac{\lambda}{1-e^{-\lambda}}\right)$-consistent. Moreover, as the Competitive Ratio will always be $\leq \frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$ for all values of η, the algorithm is $\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$-robust. □

[2] For full proof please visit the appendix

Chapter 4

Ski Rental with Multiple Predictions

In this chapter instead of using one genie, we are going to use 2 genies in order to improve and make the algorithm more robust. Furthermore, we are going to assume that one of the genies has a prediction error of 0, it always outputs the correct length of the season. The problem is that we do not know which of the two genies it is.

We also have to change the definition of consistency to:

- **Consistency:** If any of the predictions are good, then the algorithm should perform close to the best offline algorithm. We say that an algorithm is β-consistent if when $\eta = 0$ the Competitive Ratio $= \beta$.

Notation:

- y_1 is the prediction of Genie 1
- y_2 is the prediction of Genie 2

First we are going to study how to design an algorithm that takes into account both predictions.

Case 1: $(y_1 \geq b$ and $y_2 \geq b)$ or $(y_1 < b$ and $y_2 < b)$
Here both genies agree and therefore it does not matter which one is correct. If we use either prediction y_1 or y_2 in Algorithm 3, we are going to make the same decision. As the length of the season is either $x = y_1$ or $x = y_2$ then $ALG = OPT$ so Competitive Ratio $= 1$. *(Algorithm 3 is used as we completely trust one of predictions as one of the genie is always correct.)*

Case 2: $y_1 < b$ and $y_2 \geq b$:
The problem arises when the genies disagree. First, if we trust genie 1 we are going to rent until day y_1 and then we buy if the ski season continues. The worst performance of this algorithm would be when the prediction y_1 is close to b, for example, $y_1 = b - \epsilon$ (where $\epsilon > 0$ is very small) and the second genie is correct. Here the Competitive

Ratio would be $\frac{b+y_1}{b} = \frac{2b-\epsilon}{b} \approx 2$. This Competitive Ratio is the same as that of Algorithm 1 which does not use predictions, hence we have failed to improve the learning-augmented algorithm. So the reasonable decision for the algorithm to take in this case would be to buy on the first day, hence $ALG = b$. However, if in the end prediction y_1 is correct, then the Competitive Ratio is $= \frac{b}{y_1}$. The problem here is if the the length of the season is very short, our Competitive Ratio would be unbounded. So in this case it would be better to rent until day y_1 and buy after if the ski season continues yielding a Competitive Ratio of $\frac{b+y_1}{b}$.

4.1 A Deterministic Algorithm with Two Predictions

In this section we will design a deterministic algorithm that uses both predictions and is robust. Hence, our goal is to balance these two ratios so that we are covered in extreme situations.

$$\frac{b}{\gamma} = \frac{b+\gamma}{b} \tag{4.1}$$

The algorithm will work the following way:

- If there are no predictions in the interval $[0, \gamma)$ then the algorithm buys on the first day.
- If there is at least one prediction in the interval $[0, \gamma)$ and no predictions in the interval $[\gamma, b]$ then the algorithm rents until γ and buys if the season continues.
- If there is a prediction in the interval $[0, \gamma)$ and another one in the interval $[\gamma, b)$ then the algorithm rents until the end of the ski season

If we solve Equation (4.1) we get: $\gamma = \frac{-b+b\sqrt{5}}{2}$.

Algorithm 6 Deterministic algorithm with 2 machine learning models

$\gamma_0 = 0, \gamma_1 = \frac{-b+b\sqrt{5}}{2}, \gamma_2 = b$
for $i = 1$ to 2 **do**
  if there is no prediction in $[\gamma_{i-1}, \gamma_i)$ **then**
    Rent until γ_{i-1} and buy after γ_{i-1} if the season continues
    break
  end if
end for
Keep renting for all the ski season

Theorem 6. *This means that our Competitive Ratio is* $\frac{1+\sqrt{5}}{2} \approx 1.618$.

Remark. $\frac{1+\sqrt{5}}{2}$ *is known as the golden ratio and appears in a number of settings in mathematics.*

4.2 A Randomised Algorithms with Two Predictions

As done in the previous chapters we are going to try and look into randomised algorithms in order to improve our consistency bounds.

Consider that one of the predictions y_2 is greater than b and the other one y_1 is in the interval $[0, b]$. Let p be the probability we buy on the first day (that Genie 2 is correct) and $1-p$ be the probability we buy on day $y_1 = \mu$ (that Genie 1 is correct).

If y_1 is true, then $OPT = \mu$ and $\mathbb{E}[ALG] = pb + (1-p)\mu$. Hence the Competitive Ratio $= \frac{pb+(1-p)\mu}{\mu}$.

If y_2 is true, then $OPT = b$ and $\mathbb{E}[ALG] = pb+(1-p)(b+\mu)$. Hence the Competitive Ratio $= \frac{pb+(1-p)(b+\mu)}{b}$.

As before, we will balance these two Competitive Ratios so that we are covered in extreme situations

$$\frac{pb + (1-p)\mu}{\mu} = \frac{pb + (1-p)(b+\mu)}{b}$$

This gives $p = \frac{\mu^2}{\mu^2-\mu+1}$. Therefore, the Competitive Ratio for this algorithm is $\frac{b}{\mu^2-\mu+1}$ which is maximised when $\mu = \frac{b}{2}$. *We maximise the Competitive Ratio in order to account for the worst case scenario.*

Algorithm 7 Randomised Algorithm with Multiple Predictions

Let $p = \frac{b^2}{b^2-2b+4}$
if ($y_1 \leq b$ and $y_2 > b$) **then**
  Buy on the first day with probability p.
  Buy on day y_1 with probability $1-p$.
else if ($y_2 \leq b$ and $y_1 > b$) **then**
  Buy on the first day with probability $1-p$.
  Buy on day y_2 with probability p.
else if ($y_2, y_1 > b$) **then**
  Buy on the first day.
else
  Keep renting until the ski season ends
end if

Theorem 7. *The randomised algorithm of 2 experts with one having zero prediction error achieves a Competitive Ratio of* $\frac{4}{3}$.

Chapter 5

Conclusion and Results

The following is a table of the algorithms we have analysed in this dissertation. It shows how the algorithms have improved the performance of the online algorithms as they "learned" more. The consistency upper bounds got closer to 1 and the robustness lower bounds decreased. Moreover, by including multiple machine learning models the Competitive Ratio decreased. It can also be concluded that the randomised algorithms performed better than their deterministic counterparts.

Algorithm	Consistency	Robustness	Competitive Ratio
Algorithm 1			2
Algorithm 2			$\frac{e}{e-1} \approx 1.58$
Algorithm 3	1	Unbounded	
Algorithm 4	$1 < (1 + \lambda) < 2$	$2 < \frac{1+\lambda}{\lambda}$	
Algorithm 5	$1 < \frac{\lambda}{1-e^{-\lambda}} < 1.59$	$1.68 < \frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$	
Algorithm 6			$\frac{1+\sqrt{5}}{2} \approx 1.618$
Algorithm 7			$\frac{4}{3}$

From this paper it can be seen that there is still more research to be done in this field. For instance, considering more realistic cases where all of the multiple machine learning models have prediction errors.

This dissertation mainly focuses on the ski rental problem, but this problem has a number of real-life applications. For example, in the context of cloud computing (Khanafer et al. 2013) where cloud service providers (CSPs) offer to rent out resources to clients to manage their web applications.

Appendix A

Appendix

A.1 Proof of Theorem 2

Proof. Let p_k be the probability of buying on day k.
Let c be the Competitive Ratio of the randomised algorithm.
If $x < k$, then $OPT = ALG = x$.
Else, if $x \geq k$, then $OPT = b$ and $ALG = k - 1 + b$
Our goal is to minimise the Competitive Ratio c, where c is a constant such that for $\forall x \in \mathbb{N}$,

$$\mathbb{E}[ALG] \leq c \min\{b, x\}$$
$$bp_1 + (1+b)p_2 + (2+b)p_3 + ... + (x-1+b)p_x + x\sum_{j>x} p_j \leq c\min\{x, b\}$$

To simplify the proof the cost of buying the skis is given an arbitrary value, for example $b = 4$. This problem can be solved by modelling it as an infinite linear program where we want to find $p_1, p_2, p_3, ...$ such that c is minimum.
Step 1: Set up a linear program

$$\begin{aligned}
4p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + ... &\leq c && (x = 1)\\
4p_1 + 5p_2 + 2p_3 + 2p_4 + 2p_5 + 2p_6 + 2p_7 + ... &\leq 2c && (x = 2)\\
4p_1 + 5p_2 + 6p_3 + 3p_4 + 3p_5 + 3p_6 + 3p_7 + ... &\leq 3c && (x = 3)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 4p_5 + 4p_6 + 4p_7 + ... &\leq 4c && (x = 4)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 5p_6 + 5p_7 + ... &\leq 4c && (x = 5)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 9p_6 + 6p_7 + ... &\leq 4c && (x = 6)\\
&\vdots
\end{aligned}$$

Step 2: Simplify the linear program to a finite number of equations
For $x \geq 4$ the right hand side is fixed while the coefficients of the variables on the

left hand side increase. So, given that $p_k > 0$:

$$\underbrace{4p_1 + 5p_2 + 6p_3 + 7p_4 + 4p_5 + 4p_6 + 4p_7 + ...}_{x=4} \leq \underbrace{4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 5p_6 + 5p_7 + ...}_{x=5} \leq 4c$$

Hence, the 4^{th} row is deleted:

$$\begin{aligned}
4p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + ... &\leq c && (x = 1)\\
4p_1 + 5p_2 + 2p_3 + 2p_4 + 2p_5 + 2p_6 + 2p_7 + ... &\leq 2c && (x = 2)\\
4p_1 + 5p_2 + 6p_3 + 3p_4 + 3p_5 + 3p_6 + 3p_7 + ... &\leq 3c && (x = 3)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 5p_6 + 5p_7 + ... &\leq 4c && (x = 5)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 9p_6 + 6p_7 + ... &\leq 4c && (x = 6)\\
&\vdots
\end{aligned}$$

Let the optimal values be $p_4^*, p_5^* > 0$. We can set $p_4 = p_4^* + p_5^*$ and $p_5 = 0$, as the coefficient of p_4 is always smaller than or equal to the coefficient of p_5. For example, when $x = 6$:

$$4p_1 + 5p_2 + 6p_3 + 7p_4^* + 7p_5^* + 9p_6 + 6p_7 + ... = 4p_1 + 5p_2 + 6p_3 + 7p_4 + 8p_5 + 9p_6 + 6p_7 + ... \leq 4c$$

Where $4p_1 + 5p_2 + 6p_3 + 7p_4^* + 7p_5^* + 9p_6 + 6p_7 + ... \leq 4c$ is another feasible solution. This means p_5 can be removed.

$$\begin{aligned}
4p_1 + p_2 + p_3 + p_4 + p_6 + p_7 + ... &\leq c && (x = 1)\\
4p_1 + 5p_2 + 2p_3 + 2p_4 + 2p_6 + 2p_7 + ... &\leq 2c && (x = 2)\\
4p_1 + 5p_2 + 6p_3 + 3p_4 + 3p_6 + 3p_7 + ... &\leq 3c && (x = 3)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 5p_6 + 5p_7 + ... &\leq 4c && (x = 5)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 + 9p_6 + 6p_7 + ... &\leq 4c && (x = 6)\\
&\vdots
\end{aligned}$$

Doing step 2 recursively for all $k > 4$: Where the $x = k + 1$ inequality is removed as it is redundant given the $x = k$ inequality. Then setting $p_{k-1} = p_{k-1}^* + p_k^*$ and $p_k = 0$ where p_k^* and p_{k-1}^* are the optimal values. The following system is left:

$$\begin{aligned}
4p_1 + p_2 + p_3 + p_4 &\leq c && (x = 1)\\
4p_1 + 5p_2 + 2p_3 + 2p_4 &\leq 2c && (x = 2)\\
4p_1 + 5p_2 + 6p_3 + 3p_4 &\leq 3c && (x = 3)\\
4p_1 + 5p_2 + 6p_3 + 7p_4 &\leq 4c && (x \geq 4)
\end{aligned}$$

Step 3: Prove that c is the lowest possible constraint

This steps involves proving "the principle of equality", which claims that when all the constraints are tight (left hand side = to right hand side), the solution is the the

optimal solution of the linear program.

Proof by contradiction. Assume that the third inequality is slack (left hand side $\leq$ right hand side).

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &= c && (x = 1) \\ 4p_1 + 5p_2 + 2p_3 + 2p_4 &= 2c && (x = 2) \\ 4p_1 + 5p_2 + 6p_3 + 3p_4 &\leq 3c && (x = 3) \\ 4p_1 + 5p_2 + 6p_3 + 7p_4 &= 4c && (x \geq 4) \end{aligned}$$

If p_3 is increased and p_4 is decreased by $\epsilon > 0$, the equation becomes tight:

$$\begin{aligned} 4p_1 + p_2 + (p_3 + \epsilon) + (p_4 - \epsilon) &= c && (x = 1) \\ 4p_1 + 5p_2 + 2(p_3 + \epsilon) + 2(p_4 - \epsilon) &= 2c && (x = 2) \\ 4p_1 + 5p_2 + 6(p_3 + \epsilon) + 3(p_4 - \epsilon) &= 3c && (x = 3) \\ 4p_1 + 5p_2 + 6(p_3 + \epsilon) + 7(p_4 - \epsilon) &\leq 4c && (x \geq 4) \end{aligned}$$

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &= c && (x = 1) \\ 4p_1 + 5p_2 + 2p_3 + 2p_4 &= 2c && (x = 2) \\ 4p_1 + 5p_2 + 6p_3 + 3p_4 + 3\epsilon &= 3c && (x = 3) \\ 4p_1 + 5p_2 + 6p_3 + 7p_4 - \epsilon &\leq 4c && (x \geq 4) \end{aligned}$$

Equations $x = 1$ and $x = 2$ are unaffected, however equation $x = 4$ contains some slack ($-\epsilon$). p_1 is then decreased and p_4 is increased by ϵ'.

$$\begin{aligned} 4(p_1 - \epsilon') + p_2 + p_3 + (p_4 + \epsilon') &= c && (x = 1) \\ 4(p_1 - \epsilon') + 5p_2 + 2p_3 + 2(p_4 + \epsilon') &= 2c && (x = 2) \\ 4(p_1 - \epsilon') + 5p_2 + 6p_3 + 3(p_4 + \epsilon') + 3\epsilon &= 3c && (x = 3) \\ 4(p_1 - \epsilon') + 5p_2 + 6p_3 + 7(p_4 + \epsilon') - \epsilon &= 4c && (x \geq 4) \end{aligned}$$

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 - 3\epsilon' &= c && (x = 1) \\ 4p_1 + 5p_2 + 2p_3 + 2p_4 - 2\epsilon' &= 2c && (x = 2) \\ 4p_1 + 5p_2 + 6p_3 + 3p_4 - \epsilon' + 3\epsilon &= 3c && (x = 3) \\ 4p_1 + 5p_2 + 6p_3 + 7p_4 + 3\epsilon' - \epsilon &= 4c && (x \geq 4) \end{aligned}$$

Now every equation contains some slack which means that c can be decreased, a contradiction as p_1, p_2, p_3, p_4 minimise c. Hence, the resulting system is:

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &= c && (x = 1) \\ 4p_1 + 5p_2 + 2p_3 + 2p_4 &= 2c && (x = 2) \\ 4p_1 + 5p_2 + 6p_3 + 3p_4 &= 3c && (x = 3) \\ 4p_1 + 5p_2 + 6p_3 + 7p_4 &= 4c && (x \geq 4) \end{aligned}$$

Step 4: Solve the simple linear system

To find the optimal solutions each row is subtracted by the row above.

$$\begin{aligned} 4p_1 + p_2 + p_3 + p_4 &= c && (x = 1) \\ 4p_2 + p_3 + p_4 &= c && (x = 2) \\ 4p_3 + p_4 &= c && (x = 3) \\ 4p_4 &= c && (x \geq 4) \end{aligned}$$

Then each row is subtracted by the row below.

$$\begin{aligned} 4p_1 - 3p_2 &= 0 && (x = 1) \\ 4p_2 - 3p_3 &= 0 && (x = 2) \\ 4p_3 - 3p_4 &= 0 && (x = 3) \\ 4p_4 &= c && (x \geq 4) \end{aligned}$$

As p_1, p_2, p_3, p_4 are probabilities, $p_1 + p_2 + p_3 + p_4 = 1$. The optimal probabilities are then found through substitution.

Step 5: Solve the general linear system:

Going back to the general case, the same method is used as before resulting in this linear system.

$$\begin{aligned} bp_1 + p_2 + p_3 + ... + p_b &= c && (x = 1) \\ bp_2 + p_3 + ... + p_b &= c && (x = 2) \\ bp_3 + ... + p_b &= c && (x = 3) \\ &\vdots \\ bp_b &= c && (x = b) \end{aligned}$$

By carrying out the same subtraction:

$$\begin{aligned} bp_1 - (b-1)p_2 &= 0 && (x = 1) \\ bp_2 - (b-1)p_3 &= 0 && (x = 2) \\ bp_3 - (b-1)p_4 &= 0 && (x = 3) \\ &\vdots \\ bp_b &= c && (x = b) \end{aligned}$$

Thus,

$$p_j = \frac{b-1}{b} p_{j+1} \quad \text{for} \quad j \in [1, b-1] \qquad \text{and} \qquad p_b = \frac{c}{b} \quad \text{for} \quad j = b$$

So

$$p_j = \underbrace{\left[\frac{b-1}{b}\frac{b-1}{b} \cdots \frac{b-1}{b}\right]}_{\text{b - j}} \frac{c}{b}$$

Using the fact that $\sum_{j=1}^{b} p_j = 1$, we obtain $\sum_{j=1}^{b} \left(\frac{b-1}{b}\right)^{b-j} \frac{c}{b} = 1$. Hence

$$c = \frac{b}{\sum_{j=1}^{b} \left(\frac{b-1}{b}\right)^{b-j}} = \frac{b}{\sum_{i=0}^{b-1} \left(\frac{b-1}{b}\right)^{i}} = \frac{b}{\frac{1-\left(1-\frac{1}{b}\right)^{b}}{1-\left(1-\frac{1}{b}\right)}} = \frac{1-b\left(1-\frac{1}{b}\right)}{1-\left(1-\frac{1}{b}\right)^{b}} = \frac{1}{1-\left(1-\frac{1}{b}\right)^{b}}$$

As $\left(1-\frac{1}{b}\right)^{b}$ tends to $\frac{1}{e}$ as b goes to infinity, we substitute this in our equation for c to obtain

$$c \approx \frac{e}{e-1}$$

□

A.2 Proof of Theorem 4

Proof. This proof is done by cases.

Competitive Ratio $\leq \frac{1+\lambda}{\lambda}$

For $y \geq b$:

- $x < \lceil \lambda b \rceil \Rightarrow ALG = x = OPT$
- $x \geq \lceil \lambda b \rceil \Rightarrow ALG = \lceil \lambda b \rceil - 1 + b$

  Here $x = \lceil \lambda b \rceil$, gives the worst Competitive Ratio with $OPT = \lceil \lambda b \rceil$.

  So, $ALG = (\lceil \lambda b \rceil - 1) + b \leq \lambda b + b \leq \frac{1+\lambda}{\lambda} OPT$.

  Therefore

$$\text{Competitive Ratio} \leq \frac{1+\lambda}{\lambda}$$

For $y < b$:

- $x < \lceil \frac{b}{\lambda} \rceil \Rightarrow ALG = x = OPT$
- $x \geq \lceil \frac{b}{\lambda} \rceil \Rightarrow ALG = \lceil \frac{b}{\lambda} \rceil - 1 + b$

  We can see that $x = \lceil \frac{b}{\lambda} \rceil$ gives the worst Competitive Ratio, with $OPT = b$.

  So, $ALG = (\lceil \frac{b}{\lambda} \rceil - 1) + b \leq \frac{b}{\lambda} + b = \frac{1+\lambda}{\lambda} OPT$.

  Therefore

$$\text{Competitive Ratio} \leq \frac{1+\lambda}{\lambda}$$

Competitive Ratio $\leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$

For $y < b$**:**

- $x \leq b \Rightarrow ALG = x = OPT$

- $x > \lceil \frac{b}{\lambda} \rceil \Rightarrow ALG = (\lceil \frac{b}{\lambda} \rceil - 1) + b, OPT = b$
- $b < x \leq \lceil \frac{b}{\lambda} \rceil \Rightarrow ALG = x, OPT = b$

  From the second bullet point:

  $$ALG = (\lceil \frac{b}{\lambda} \rceil - 1) + b \leq \frac{b}{\lambda} + b < b + \frac{1}{1-\lambda}\eta < OPT + \frac{1}{1-\lambda}\eta < (1+\lambda)OPT + \frac{1}{1-\lambda}\eta$$

  Therefore

  $$\text{Competitive Ratio} \leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$$

  From the third bullet point:

  $$ALG = x = b + (x - b) \leq OPT + (x - y) = OPT + \eta < (1+\lambda)OPT + \frac{1}{1-\lambda}\eta$$

  Therefore

  $$\text{Competitive Ratio} \leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$$

For $y \geq b$:

- $x < \lceil \lambda b \rceil \Rightarrow ALG = x = OPT$
- $\lceil \lambda b \rceil < x < b \Rightarrow ALG = (\lceil \lambda b \rceil - 1) + b,\ OPT = x$
- $x \geq b \Rightarrow ALG = (\lceil \lambda b \rceil - 1) + b,\ OPT = b$

  From the second bullet point:

  $$ALG = (\lceil \lambda b \rceil - 1) + b < \lambda b + b = (1+\lambda)b \leq (1+\lambda)y = (1+\lambda)(x + (y - x))$$
  $$\leq (1+\lambda)(OPT + \eta) = (1+\lambda)OPT + (1+\lambda)\eta \leq (1+\lambda)OPT + \frac{1}{1-\lambda}\eta$$

  Therefore

  $$\text{Competitive Ratio} \leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$$

  From the third bullet point:

  $$ALG = (\lceil \lambda b \rceil - 1) + b \leq \lambda b + b = (1+\lambda)OPT \leq (1+\lambda)OPT + \frac{1}{1-\lambda}\eta$$

  Therefore

  $$\text{Competitive Ratio} \leq (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$$

Hence we have shown that the Competitive Ratio is at most $\frac{1+\lambda}{\lambda}$ or $(1+\lambda) + \frac{\eta}{(1-\lambda)OPT}$ depending on which one is the smallest. The algorithm is $\frac{1+\lambda}{\lambda}$-robust as the Competitive Ratio will always be smaller than $\frac{1+\lambda}{\lambda}$ for all η. The algorithm is $(1+\lambda)$-consistent as the Competitive Ratio $= (1+\lambda)$ when $\eta = 0$. □

A.3 Proof of Theorem 5

Proof. We are going to consider the different cases of the values of y and x.

1. $\underline{y \geq b, x \geq k}$

$OPT = \min\{b, x\}$ and $ALG = b + i - 1$ for day i.

$$\begin{aligned}
\mathbb{E}[ALG] &= \sum_{i=1}^{k}(b+i-1)q_i = \sum_{i=1}^{k}(b+i-1)\left(\frac{b-1}{b}\right)^{k-i}\frac{1}{b(1-(1-\frac{1}{b})^k} \\
&= \sum_{i=1}^{k}\frac{(b+i-1)}{b}\left(\frac{b-1}{b}\right)^{k-i}\frac{1}{1-\left(1-\frac{1}{b}\right)^k} \\
&= \frac{k}{1-\left(1-\frac{1}{b}\right)^k} \leq \frac{k}{1-e^{\frac{-k}{b}}} \leq \left(\frac{\frac{k}{b}}{1-e^{\frac{-k}{b}}}\right)(OPT+\eta) \\
&\leq \left(\frac{\lambda}{1-e^{-\lambda}}\right)(OPT+\eta)
\end{aligned}$$

2. $\underline{y \geq b, x < k}$

$OPT = x$ and $ALG = b + i - 1$ for day $i \leq x$.

$$\begin{aligned}
\mathbb{E}[ALG] &= \sum_{i=1}^{x}(b+i-1)q_i + \sum_{i=x+1}^{k} xq_i \\
&= \frac{1}{b(1-(1-\frac{1}{b})^k}\left[\sum_{i=1}^{x}(b+i-1)\left(\frac{b-1}{b}\right)^{k-i} + \sum_{i=x+1}^{k} x\left(\frac{b-1}{b}\right)^{k-i}\right] \\
&= \frac{bx}{b(1-(1-\frac{1}{b})^k} \leq \frac{OPT}{1-e^{-\frac{k}{b}}} \leq \frac{OPT}{1-e^{-(\lambda-\frac{1}{b})}}
\end{aligned}$$

The inequality above gives us an upper bound for the Competitive Ratio. However, we can also derive the following inequality that includes the prediction error.

$$\begin{aligned}
\mathbb{E}[ALG] &\leq \frac{x}{1-e^{-\frac{k}{b}}} = \frac{OPT}{1-e^{-\frac{k}{b}}} = \left(\frac{\frac{k}{b}}{1-e^{-\frac{k}{b}}}\right)OPT + \left(\frac{\frac{(b-k)}{b}}{1-e^{-\frac{k}{b}}}\right)x \\
&\leq \left(\frac{\frac{k}{b}}{1-e^{-\frac{k}{b}}}\right)OPT + \left(\frac{\frac{\eta}{b}}{1-e^{-\frac{k}{b}}}\right)k \leq \left(\frac{\lambda}{1-e^{-\lambda}}\right)(OPT+\eta)
\end{aligned}$$

3. $\underline{y < b, x < l}$

$OPT = \min\{b, x\}$

$$\begin{aligned}
\mathbb{E}[ALG] &= \sum_{i=1}^{x}(b+i-1)r_i + \sum_{i=x+1}^{l} xr_i = \frac{x}{1-\left(1-\frac{1}{b}\right)^l} \\
&\leq \left(\frac{1}{1-e^{-\frac{l}{b}}}\right)x \leq \left(\frac{\lambda}{1-e^{-\lambda}}\right)(OPT+\eta)
\end{aligned}$$

4. $\underline{y < b, x \geq l}$

$OPT = b$

$$\begin{aligned}
\mathbb{E}[ALG] &= \sum_{i=1}^{x}(b+i-1)r_i = \frac{l}{1-\left(1-\frac{1}{b}\right)^l} \leq \frac{\lceil\frac{b}{\lambda}\rceil}{1-e^{-\frac{l}{b}}} \leq \left(\frac{\frac{1}{\lambda}+\frac{1}{b}}{1-e^{-\frac{1}{\lambda}}}\right)OPT \\
&\leq \left(\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}\right)OPT
\end{aligned}$$

The inequality above gives us an upper bound for the Competitive Ratio. However, we can also derive the following inequality that includes the prediction error.

$$\mathbb{E}[ALG] = \frac{l}{1-\left(1-\frac{1}{b}\right)^{l}} \leq \frac{l}{1-e^{\frac{-l}{b}}} \leq \left(\frac{\lambda}{1-e^{-\lambda}}\right)(OPT+\eta)$$

We can see that from the cases the Competitive Ratio is at most $\left(\frac{\lambda}{1-e^{-\lambda}}\right)(1+\frac{\eta}{OPT})$ or $\left(\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}\right)$. Hence, when $\eta = 0$ the Competitive Ratio $= \frac{\lambda}{1-e^{-\lambda}}$ so the algorithm is $\left(\frac{\lambda}{1-e^{-\lambda}}\right)$-consistent. Moreover, as the Competitive Ratio will always be $\leq \frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$ for all values of η, the algorithm is $\frac{1}{1-e^{-(\lambda-\frac{1}{b})}}$-robust. □

Bibliography

Anand, K., Ge, R. & Panigrahi, D. (2020), Customizing ML Predictions for Online Algorithms, in 'International Conference on Machine Learning', PMLR, pp. 303–313.

Borodin, A. & El-Yaniv, R. (1998), 'Online Computation and Competitive Analysis'.

Gollapudi, S. & Panigrahi, D. (2019), Online Algorithms for Rent-or-Buy with Expert Advice, in 'International Conference on Machine Learning', PMLR, pp. 2319–2327.

Karlin, A. R., Kenyon, C. & Randall, D. (2003), 'Dynamic TCP Acknowledgment and Other Stories about e/(e- 1)', Algorithmica **36**, 209–224.

Karlin, A. R., Manasse, M. S., McGeoch, L. A. & Owicki, S. (1994), 'Competitive Randomized Algorithms for Nonuniform Problems', Algorithmica **11**(6), 542–571.

Karlin, A. R., Manasse, M. S., Rudolph, L. & Sleator, D. D. (1988), 'Competitive Snoopy Caching', Algorithmica **3**(1), 79–119.

Khanafer, A., Kodialam, M. & Puttaswamy, K. P. (2013), The Constrained Ski-Rental Problem and its Application to Online Cloud Cost Optimization, in '2013 Proceedings IEEE INFOCOM', IEEE, pp. 1492–1500.

Kodialam, R. (2014), 'Competitive Algorithms for an Online Rent or Buy Problem with Variable Demand', SIAM Undergraduate Research Online **7**, 233–245.

Kumar, R., Purohit, M. & Svitkina, Z. (2018), Improving Online Algorithms via ML Predictions, in 'Proceedings of the 32nd International Conference on Neural Information Processing Systems', pp. 9684–9693.

Lotker, Z., Patt-Shamir, B. & Rawitz, D. (2008), 'Rent, Lease or Buy: Randomized Algorithms for Multislope Ski Rental', arXiv preprint arXiv:0802.2832 .

Lykouris, T. & Vassilvtiskii, S. (2018), Competitive Caching with Machine Learned Advice, in 'International Conference on Machine Learning', PMLR, pp. 3296–3305.

Medina, A. M. & Vassilvitskii, S. (2017), 'Revenue Optimization with Approximate Bid Predictions', arXiv preprint arXiv:1706.04732 .

Meyerson, A. (2005), The Parking Permit Problem, in '46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)', IEEE, pp. 274–282.

Wei, A. & Zhang, F. (2020), 'Optimal Robustness-Consistency Trade-offs for Learning-Augmented Online Algorithms', arXiv preprint arXiv:2010.11443 .