Customizing ML Predictions for Online Algorithms

Keerti Anand[1] **Rong Ge**[1] **Debmalya Panigrahi**[1]

Abstract

A popular line of recent research incorporates ML advice in the design of online algorithms to improve their performance in typical instances. These papers treat the ML algorithm as a black-box, and redesign online algorithms to take advantage of ML predictions. In this paper, we ask the complementary question: can we redesign ML algorithms to provide better predictions for online algorithms? We explore this question in the context of the classic rent-or-buy problem, and show that incorporating optimization benchmarks directly in ML loss functions leads to significantly better performance, while maintaining a worst-case adversarial result when the advice is completely wrong. We support this finding both through theoretical bounds and numerical simulations, and posit that "learning for optimization" is a fertile area for future research.

1. Introduction

Optimization under uncertainty is a classic theme in the fields of algorithm design and machine learning. In the former, the framework of online algorithms adopts a conservative approach and optimizes for the worst case (or adversarial) future. While this ensures robustness, the inherent pessimism of the adversarial approach often results in weak guarantees. Machine learning (ML), on the other hand, takes a more optimistic approach of trying to predict the future by fitting an appropriate model to past data. Indeed, a popular line of recent research is to incorporate ML advice in the design of online algorithms to improve their performance while preserving the inherent robustness of the framework (see related work for references). In this line of research, ML is used as a *black box*, and the focus is on redesigning online algorithms to use predictions generated by any ML technique. In this paper, we ask the complementary question: *can we re-design learning algorithms to better serve optimization objectives?*

The key to this question is the observation that unlike in a generic learning setting, we are not interested in traditional loss functions such as classification error or mean-squared loss, but only in the eventual performance of the online algorithm. The performance of the online algorithm is measured by its *competitive ratio* – the ratio between the cost of the algorithm and the offline optimal solution. By leveraging predictions from ML algorithms, one can hope to achieve a better competitive ratio in the typical case. Even if the ML algorithm does not make accurate predictions, it suffices if the learning errors do not adversely affect the decisions taken by the (online) optimization algorithm. Instead of treating the learning algorithm and the subsequent optimization as independent modules as in the previous line of work, we ask if we can improve the overall online algorithm by designing them in conjunction. That is, we seek to design a learning algorithm specific to the optimization task at hand, and an optimization algorithm that is aware of the learning algorithm that generated the predictions.

We investigate this question in the context of the classic *rent-or-buy* (or *ski rental*) problem. In this problem, the algorithm is faced with one of two choices: a small recurring (rental) cost that it pays throughout, or a large (buying) cost upfront but no cost thereafter. This choice routinely arises in our daily lives, such as in the decision to rent or buy a house, as also in corporate decisions to rent or buy data centers, expensive equipment, and so on. Naturally, the optimal choice depends on the duration of use, a longer duration justifying the decision to buy instead of renting. But, this is where the uncertainty lies: the length of use is often not known in advance. The ski rental problem is perhaps the most fundamental, and structurally simplest, of all problems in online algorithms, and has been widely studied in many contexts (see, e.g., Karlin et al. (1994; 2003); Lotker et al. (2008); Khanafer et al. (2013); Kodialam (2014)), including that of online algorithms with ML predictions (Purohit et al., 2018; Gollapudi & Panigrahi, 2019). We formally define this problem next.

The ski rental problem. In the ski rental problem, a skier has two options: to buy skis at a one time cost of $\$B$ or to rent them at a cost of \$1 per day. The skier does not know the length of the ski season in advance, and only learns it once the season ends. Note that if the length of the season were known, then the optimal policy is to buy at the beginning of the season if it lasts longer than B days, and rent every day if it is shorter. But, in the absence of this information, an algorithm has to decide the duration of renting skis before buying them. It is well-known that the best competitive ratio achievable by a deterministic algorithm for this problem is 2 (e.g., Karlin et al. (1988)), and that by a randomized algorithm is $\frac{e}{e-1}$ (e.g., Karlin et al. (1994)). The ski-rental problem (Karlin et al., 1994; Lotker et al., 2008; Khanafer et al., 2013; Kodialam, 2014), and variants such as TCP acknowledgment (Karlin et al., 2003), the parking permit problem (Meyerson, 2005), snoopy caching (Karlin et al., 1988), etc. model the fundamental difficulty in decision making under uncertainty in many situations.

The learning framework. We use a classic PAC learning framework. Namely, the learning algorithm observes feature vectors $x \in \mathbb{R}^d$ comprising, e.g., weather predictions, skier history, etc. and aims to predict scalars $y \in \mathbb{R}^+$ denoting the length of the ski season. We assume that (x, y) belongs to an unknown joint distribution $\mathbb{K}$. The learning algorithm observes n samples (the "training set") from $\mathbb{K}$. Typically, these samples would be used to train a model that maps feature vectors x to predictions $\tilde{y} = f(x)$ that minimizes some loss function (e.g., mean squared error, hinge loss, etc.) defined on $\mathbb{K}$. In our problem, however, the goal is not to predict the unknown y, but rather to optimize the solution to the ski rental instance defined by y. Consequently, the learning algorithm skips y altogether and outputs a solution to the optimization problem directly. For the ski rental problem, this amounts to defining a function $\theta(x)$ that maps the feature vector x to the duration of renting skis. The expected competitive ratio is then given by the competitive ratio of this policy $\theta(x)$ defined on distribution $\mathbb{K}$. We call this a "learning-to-rent" algorithm.

Our Contributions. Our goal is to design a learning-to-rent algorithm with a competitive ratio of $(1 + \varepsilon)$, and analyze the dependence of the number of samples n on the value of ε. Contrast this with online algorithms for this problem that can at best achieve a competitive ratio of $\frac{e}{e-1}$ (e.g., Karlin et al. (1994)). If the joint distribution (x, y) is arbitrary, then one cannot hope to achieve a competitive ratio of $(1 + \varepsilon)$ since every sample may have a different x and the conditional distributions $y|x$ can be unrelated for different values of x. However, it is natural to assume that the joint distribution on (x, y) is **Lipschitz** in the sense that nearby values of x imply similar conditional distributions $y|x$. Our first contribution (Theorem 2) is to design a learning-to-rent algorithm whose competitive ratio is within a factor of $(1 + \varepsilon)$ of the best bound for competitive ratio (possible for that distribution), under only the Lipschitz assumption. First, we discretize the domain of x using an ϵ-net. Then, for each cell in the ϵ-net, we have one of two cases. Either, there are sufficiently many samples to estimate the conditional distribution $y|x$, or a baseline online algorithm can be used for the cell if it has very few samples. The dependence of the number of samples n on the number of dimensions d is exponential, which we show is indeed necessary (Theorem 3).

Our next goal is to improve the dependence on d since the number of features in a typical setting can be rather large, which would make the previous algorithm prohibitively expensive. To this end, we use a PAC learning approach to address the problem. Since the optimal ski rental policy exhibits threshold behavior (rent throughout if $y < B$ and buy at the outset if $y \geq B$), we treat the underlying learning problem as a classification task. In particular, we introduce an auxiliary binary variable z that captures the two regimes for the optimal ski rental policy:

$$z = \begin{cases} 1 & \text{if } y \geq B \\ 0 & \text{if } y < B \end{cases}$$

Our first result is that if z belongs to a concept class that is (ε, δ) PAC-learnable from x, then we can obtain a learning-to-rent algorithm that achieves a competitive ratio of $(1 + 2\sqrt{\epsilon})$ with probability $1 - \delta$. This implies, for instance, that if there were a linear classifier for z, then the number of samples n can be decreased from exponential in d to linear in d, specifically $O(d/\epsilon)$. While this is a significant improvement, we hope to do even better by exploiting the specific structure of the ski rental problem. In particular, we observe that the classification error is almost entirely due to samples close to the threshold, but for values of y close to B, mis-classifying z does not cost us significantly in the ski rental objective. This allows us to create an artificial margin around the classification boundary and discard all samples that appear in this margin. Using this improvement, we can improve the sample complexity of the training set to remove all dependence on d (although at a slightly worse dependence on ε). We also consider a noisy model where the labels in the training set are noisy and derive the dependence of the competitive ratio of the algorithm on the noise rate. Finally, we perform numerical simulations to evaluate our learning-to-rent policies in typical settings. These empirical results provide further justification that incorporating the optimization objective in the learning algorithm leads to significant improvements.

Related Work. A robust literature is beginning to emerge in incorporating ML predictions in online algorithms to obtain optimistic bounds if the predictions are correct, but preserve the robustness of worst-case competitive analysis for inaccurate predictions. This model has been applied to a wide variety of problems including auction pricing (Medina & Vassilvitskii, 2017), ski rental (Purohit et al., 2018; Gollapudi & Panigrahi, 2019), caching (Lykouris & Vassilvitskii, 2018; Rohatgi, 2020; Jiang et al., 2020), scheduling (Purohit et al., 2018; Lattanzi et al., 2020; Mitzenmacher, 2020), frequency estimation (Hsu et al., 2019), Bloom filters (Mitzenmacher, 2018), etc. As described earlier, these results consider ML as a black box and re-design the online algorithm, whereas we take the complementary approach of re-designing the learning algorithm to suit the optimization task.

Our main idea is to modify the loss function in the learning algorithm to incorporate the optimization objective. There has been previous research in a similar spirit, where the loss function in learning is adapted to suit specific purposes, albeit different ones from our work. For instance, Huang et al. (2019) give an "Adaptive Loss Alignment" scheme to meta-learn the loss function to directly optimize the evaluation metric in the context of Reinforcement Learning. Gupta & Roughgarden (2017) present a framework for algorithm selection as a statistical learning problem. This framework captures, for instance, the notion of "self-improving algorithms", where the goal is to learn the input distribution and adaptively design an optimal policy Ailon et al. (2011). A related line of research, pioneered by Cole & Roughgarden (2014), is that of optimizing on samples of the input rather than the entire input (see also Morgenstern & Roughgarden (2016); Balkanski et al. (2016; 2017)). Yet another example of adapting the loss function in learning is in Cost Sensitive Learning (Elkan, 2001), where mis-classication errors incur non-uniform penalties (see also Kamalaruban & Williamson (2018); Ling & Sheng (2008)).

2. Preliminaries

For notational convenience, we consider a continuous version of the ski rental problem, where the buying cost is \$1, and the length of the ski season is denoted by $y \in \mathbb{R}^+$. Therefore, the optimal offline algorithm is to buy when $y \geq 1$ and rent when $y < 1$. (These assumptions are w.l.o.g. by appropriate scaling.) We also denote the feature vector by $x \in \mathbb{R}^d$ (e.g., weather predictions, skier behavior, etc.) and assume that (x, y) belongs to an unknown joint distribution $\mathbb{K}$. Given a feature vector x, the goal of the algorithm is to produce a threshold $\theta(x)$ such that the skier rents till time $\theta(x)$ and buys at that point if the ski season is longer. We call $\theta(x)$ the *wait time* of the algorithm.

If the distribution $\mathbb{K}$ were known to the algorithm, then for each input x, it can compute the conditional distribution $y|x$ and solve the resulting *stochastic* ski rental problem, i.e., where the input is drawn from a given distribution. It is well known that the optimal strategy in this case can we described by a fixed wait time that we denote $\theta^*(x)$.

Of course, in general, the distribution $\mathbb{K}$ is not known to the algorithm, and has to be "learned" from training data. The "learning-to-rent" algorithm observes n training samples $(x_i, y_i) \sim \mathbb{K}$, and based on them, generates a function $\theta(x)$ that maps feature vectors x to the wait time. The (expected) competitive ratio of the algorithm is given by:

$$\textsc{cr}(\theta, \mathbb{K}) = \mathbb{E}_{(x,y)\sim\mathbb{K}}[g(\theta(x), y)] \qquad (1)$$

$$\text{where } g(\theta(x), y) = \begin{cases} \frac{y}{\min\{y,1\}} & \text{when } y < \theta(x) \\ \frac{1+\theta(x)}{\min\{y,1\}} & \text{when } y \geq \theta(x). \end{cases} \qquad (2)$$

The goal of the learning-to-rent algorithm is to output a function $\theta(\cdot)$ that minimizes $\textsc{cr}$ in Eq. (1). Since the ideal strategy is to output the function $\theta^*(\cdot)$, we measure the performance of the algorithm as the ratio between $\textsc{cr}(\theta, \mathbb{K})$ and $\textsc{cr}(\theta^*, \mathbb{K})$.

Definition 1. *A learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is said be (ϵ, δ)-accurate with n samples, if for any distribution $\mathbb{K}$, after observing n samples, we have the following guarantee with probability (over the samples) of at least $1 - \delta$:*

$$\textsc{cr}(\theta, \mathbb{K}) \leq (1 + \epsilon) \cdot \textsc{cr}(\theta^*, \mathbb{K}). \qquad (3)$$

If we say that an algorithm is $(1 + \epsilon)$-accurate, we mean Eq. (3) holds for some fixed constant δ.

We also make use of the notion of "robustness" defined in (Purohit et al., 2018):

Definition 2. *A learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is said to be γ-robust if $g(\theta(x), y) \leq \gamma$ for any feature x and any ski-length y.*

Lemma 1. *The learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is $\left(1 + \frac{1}{\theta_0}\right)$ robust where $\theta_0 = \min_{x\in\mathbb{R}^d} \theta(x)$*

Proof. Note that the function $g(\theta, y)$ achieves its maximum value at $y = \theta + \rho$ where $\rho \to 0^+$. In this case, the algorithm pays $1 + \theta$, while the optimal offline cost is θ. This gives us that $\max_{y\in\mathbb{R}^+} g(\theta, y) = \left(1 + \frac{1}{\theta}\right)$. Lastly, since there is no x such that $\theta(x) < \theta_0$, therefore $\max_{y\in\mathbb{R}^+, x\in\mathbb{R}^d} g(\theta(x), y) \leq \left(1 + \frac{1}{\theta_0}\right)$. ☐

3. A General Learning-to-Rent Algorithm

As described in the introduction, it is natural (and required) to assume that the joint distribution $\mathbb{K}$ on (x, y) is **Lipschitz** in the sense that similar feature vectors x imply similar conditional distributions $y|x$. In this section, our main contribution is to design a learning-to-rent algorithm under this minimal assumption.

First, we give the precise definition of the Lipschitz property we require. In particular, we measure distances between distributions using the *earth mover distance* ($\textsc{emd}$) metric.

Definition 3. *For probability distributions $\mathbb{X}, \mathbb{Y}$ over $\mathbb{R}^d$,*

$$\textsc{emd}(\mathbb{X}, \mathbb{Y}) = \min_{\mathbb{K}:\mathbb{K}|x=\mathbb{X}, \mathbb{K}|y=\mathbb{Y}} \left(\mathbb{E}_{(x,y)\sim\mathbb{K}}[||x - y||]\right).$$

The joint distribution $\mathbb{K}$ above is such that its marginals with respect to y and x are equal to $\mathbb{X}$ and $\mathbb{Y}$ respectively.

We now define the Lipschitz property using $\textsc{emd}$ as the distance measure between distributions.

Definition 4. *A joint distribution on* $(x, y) \in \mathbb{R}^d \times \mathbb{R}^+$ *is said to be L-Lipschitz iff for all* $x_1, x_2 \in \mathbb{R}^d$*, the marginal distributions* $\mathbb{Y}_1 = y|x_1$, $\mathbb{Y}_2 = y|x_2$ *satisfy* $\text{EMD}(\mathbb{Y}_1, \mathbb{Y}_2) \le L \cdot \|x_1 - x_2\|_2$.

Now we are ready to state our main result in this section: a learning-to-rent algorithm with optimal sample complexity.

Theorem 2. *For a learning-to-rent problem, if* $x \in [0,1]^d$*, and the joint distribution* (x, y) *is L-Lipschitz, then there exists an algorithm that uses* $n = \left(\frac{L\sqrt{d}}{\epsilon}\right)^{O(d)}$ *samples and is* $(1+\epsilon)$*-accurate with high probability* [1].

Algorithm 1 Outputs θ_A for a given distribution on y

```
Query $\left(\frac{\delta}{\epsilon^6}\right)$ samples for some constant $\delta > 0$.
Initialize array $l$ of length $\frac{1}{\epsilon^2}$
Let $\ell[\theta] \leftarrow$ average of $g(\theta, y)$ over all samples $y$.
return $\theta_A \leftarrow \arg\min_{\theta \in [\epsilon, 1/\epsilon], \theta/\epsilon \in \mathbb{N}} \ell[\theta]$.
```

Let us first consider the simple case where we have a fixed x and only consider the conditional distribution $y|x$. In this case, it is natural to optimize θ on the empirical samples of y. However, if we don't put any constraint on θ, the competitive ratio for a sample y can be unbounded (this can happen when θ is close to 0 or very large), which might hurt generalization. We solve this problem by proving that it suffices to consider θ in the range $[\epsilon, 1/\epsilon]$ in order to get an $(1+\epsilon)$-accurate solution. (See Algorithm 1.)

Algorithm 2 Outputs $\theta_A(x)$ for multi-dimensional x

```
Divide the hyper-cube $[0,1]^d$ into sub-cubes of side length $\frac{\epsilon^3}{64L\sqrt{d}}$ each. The number of such cubes is $N = \left(\frac{64L\sqrt{d}}{\epsilon^3}\right)^d$.
Index the cubes by $i$, where $1 \le i \le N$.
Query $\Pi = \left(\frac{1024L\sqrt{d}}{\epsilon^6}\right)^{2d}$ samples, and let $I_\epsilon = [\epsilon, 1/\epsilon]$.
Set threshold $\tau = \left(\frac{64L\sqrt{d}}{\epsilon^8}\right)^d$.
for each sub-cube $C_i$:
  if the number of samples from the sub-cube exceeds $\tau$ then
    Compute $\theta_i \leftarrow \arg\min_{\theta \in I_\epsilon, \theta/\epsilon \in \mathbb{N}} \mathbb{E}_{(x,y):x \in C_i}[g(\theta, y)]$.
    For all $x \in C_i$: return $\theta_A(x) \leftarrow \theta_i$.
  else For all $x \in C_i$: return $\theta_A(x) \leftarrow 1$.
```

To go from a single x to the whole distribution, the main idea is to discretize the domain of x using an ϵ-net for small enough ϵ.[2] For each cell in the ϵ-net, we show that if there are enough samples in the training set from that cell, then we can estimate the conditional probability $y|x$ to a sufficient degree of accuracy for the optimization loss to be bounded by $1+\epsilon$. On the other hand, if there are too few samples, then the probability density in the cell is small enough that it suffices to use a worst case online algorithm for all test data in the cell. (The formal algorithm is given in Algorithm 2.) We defer the formal analysis of this algorithm, which establishes Theorem 2, to the appendix.

The main shortcoming of the above result is that there is an exponential dependence of the sample complexity on the number of feature dimensions d. Unfortunately, this dependence is necessary, as shown by the next theorem, whose proof also appears in the appendix.

Theorem 3. *For any learning-to-rent algorithm, there exists a family of* 1*-Lipschitz joint distributions* (x, y) *where* $x \in [0,1]^d$ *such that the algorithm must query* $\frac{1}{\epsilon^{\Omega(d)}}$ *samples in order to be* $(1+\epsilon)$*-accurate, for small enough* $\epsilon > 0$.

4. A PAC Learning Approach to the Learning-to-Rent Problem

In the previous section, we saw that without making further assumptions, the number of samples required by a learning-to-rent algorithm will be exponential in the dimension of the feature space. To avoid this, we try to identify reasonable assumptions that allow the learning-to-rent algorithm to be more efficient.

We follow the traditional framework of PAC learning. Recall that in PAC learning, the true function between features and labels is restricted to be in a certain function class $\mathcal{C}$:

Definition 5. *Consider a set* $X \in \mathbb{R}^d$ *and a concept class* $\mathcal{C}$ *of Boolean functions* $X \rightarrow \{0,1\}$*. Let there be an arbitrary hypothesis* $c \in \mathcal{C}$*. Let* P *be a PAC learning algorithm that takes as input the set* S *comprising* m *samples* (x_i, y_i) *where* x_i *is sampled from a distribution* $\mathbb{D}$ *on* X *and* $y_i = c(x_i)$*, and outputs a hypothesis* $\hat{c}$*.* P *is said to be have* ϵ *error with failure probability* δ*, if with probability at least* $1-\delta$ *(over the samples in* S*):*

$$\mathbb{P}_{x \sim \mathbb{D}}[\hat{c}(x) \neq c(x)] \le \epsilon.$$

Standard results in learning theory show that if the function class $\mathcal{C}$ is simple, the PAC-learning problem can be solved with few samples. In the learning-to-rent problem, our goal is to learn the optimal policy $\theta^*(\cdot)$.

We consider the situation where the value of y is deterministic given x. This assumption basically says that the features contain enough information to predict the length of the ski season.

Assumption 1. *In the input distribution* $(x, y) \sim \mathbb{K}$ *for the learning-to-rent algorithm, the value of* y *is a deterministic function of* x *(*$y = f(x)$ *for some function* f*).*

---

[1] with probability exceeding $1 - \epsilon^{\Omega(d)}$

[2] The ϵ in the ϵ-net is not the same as the accuracy parameter ϵ. We are overloading ϵ in this description because the reader may be familiar with the term ϵ-net; in the formal algorithm (Algorithm 2), we avoid this overloading.

Note that in this case, the optimal solution is going to have competitive ratio of 1, so an $(1+\epsilon)$-accurate learning-to-rent algorithm must have competitive ratio $1+\epsilon$.

Because of Assumption 1, the entire feature space can be divided into two regions: one where $y < 1$ and renting is optimal, and the other where $y \geq 1$ and buying at the outset is optimal. If the boundary between these two regions is PAC-learnable, we can hope to improve on the result from the previous section. This could also be seen as the weaker version of Assumption 1:

Assumption 2. *In the input distribution $(x, y) \sim \mathbb{K}$ for the learning-to-rent algorithm where X is the domain for x, there exists a hypothesis $c : X \mapsto \{0, 1\}$ lying in a concept class $\mathcal{C}$ such that c separates the region $y \geq 1$ and $y < 1$. For notational purposes, we say $c(x) = 1$ when $y \geq 1$ and $c(x) = 0$ when $y < 1$.*

PAC-learning as a black box. We first show that in this setting, one can use learning algorithm as a black-box: if we can PAC learn the concept class $\mathcal{C}$ accurately, then we can get a competitive algorithm for the ski-rental problem. The precise algorithm is given in Algorithm 3. This only uses the weaker assumption 2, that only requires the binary variable $(y > 1)$ to be deterministic in x.

Algorithm 3 Black box learning-to-rent algorithm

```
Set $\tau = \sqrt{\epsilon}$
Learning: Query $n$ samples. Train a PAC-learner.
For test input $x$: if PAC-learner predicts $y \geq 1$, then $\theta(x) = \tau$, else $\theta(x) = 1$.
```

The next theorem determines the sample complexity of Algorithm 3. The proof is deferred to the appendix.

Theorem 4. *Given an algorithm that PAC learns the concept class $\mathcal{C}$ with error ϵ and failure probability δ, there exists a learning-to-rent algorithm that has a competitive of $(1 + 2\sqrt{\epsilon})$ with probability $1 - \delta$. Moreover, this algorithm is $\left(1 + \frac{1}{\sqrt{\epsilon}}\right)$ robust.*

Remark 1. *The above analysis can be refined for asymmetrical errors (where the classification errors on the two sides are different) essentially showing that the algorithm is more sensitive to errors of one type than the other. Further details on this appear in the appendix.*

Next, we show that the relationship between PAC-learning and learning-to-rent, established in one direction in Theorem 4, actually holds in other direction too. In other words, we can derive a PAC-learning algorithm from a learning-to-rent algorithm. This implies, for instance, that existing lower bounds for PAC learning also apply to learning-to-rent algorithms. Therefore, in principle, the sample complexity of the algorithm in Theorem 4 is (near) optimal without any further assumptions. The proof of this theorem is deferred to the appendix.

Theorem 5. *If there exists an (ϵ, δ)-accurate learning-to-rent algorithm for a concept class $\mathcal{C}$ with n samples, then there exists an $O(\epsilon, \delta)$-PAC learning algorithm for $\mathcal{C}$ with the same number of samples.*

Margin-based PAC-learning for data without margin. Theorem 4 is very general as there are many concept classes which we know how to PAC-learn. On the other hand, even for a simple linear separator, PAC-learning requires at least $\Omega(d)$ samples in d dimensions, which can be costly for large d. In learning theory, the number of samples can be reduced when the VC-dimension of the concept class is small:

Theorem 6 (e.g., (Kearns et al., 1994))**.** *A concept class of VC-dimension D is (ϵ, δ) PAC-learnable using $n = \Theta\left(\frac{D+\log(1/\delta)}{\epsilon}\right)$ samples. For fixed δ, the sample complexity of PAC-learning is $\Theta\left(\frac{D}{\epsilon}\right)$.*

In particular, this result is used when the underlying data distribution has a *margin*, which is the distance of the closest point to the decision boundary:

Definition 6. *Given a data set $D \in \mathbb{R}^d \times \{0, 1\}$ and a separator c, the margin of D with respect to c is defined as $\min_{x' \in \mathbb{R}^d, (x,y) \in D, c(x') \neq y} \|x - x'\|$.*

The advantage of having a large margin is that it reduces the VC-dimension of the concept class. Since the precise dependence of the VC-dimension on the width of the margin (denoted α) depends on the concept class $\mathcal{C}$, let us denote the VC-dimension by $D(\alpha)$.

Crucially, we will show that in the learning-to-rent algorithm, it is possible to *reduce the sample complexity even if the original data $(x, y) \sim \mathbb{K}$ does not have any margin!* The main idea is that the learning-to-rent algorithm can ignore training data in a suitably chosen margin since $y \approx 1$ for points in the margin, and the competitive ratio of the optimization algorithm is close to 1 for these points even with no additional information. Thus, although the algorithm fails to learn the label of test data near the margin reliably, this does not significantly affect the eventual competitive ratio of the learning-to-rent algorithm.

Note that the L-Lipschitz property under Assumption 1 is:

Assumption 3. *For $x_1, x_2 \in X$ where X is the domain of x, if $y_1 = f(x_1)$ and $y_2 = f(x_2)$, we have $|y_1 - y_2| \leq L \cdot \|x_1 - x_2\|$.*

We now give a learning-to-rent algorithm that uses this margin-based approach (Algorithm 4). Recall that α is the width of the margin used by the algorithm; we will set the value of α later.

The filtering process creates an artificial margin:

Algorithm 4 Margin-based learning-to-rent algorithm

```
Set $\gamma = L\alpha$.
Learning: Query $n$ samples. Discard samples $(x_i, y_i)$ where $y_i \in [1-\gamma, 1+\gamma]$. Use the remaining samples to train a PAC-learner with margin $\alpha$.
For test input $x$: if PAC-learner predicts $y \geq 1$, then $\theta(x) = \gamma$, else $\theta(x) = 1+\gamma$.
```

Lemma 7. *In Algorithm 4, the samples used in the PAC learning algorithm have a margin of α.*

We now analyze the sample complexity of Algorithm 4.

Theorem 8. *Given a concept class $\mathcal{C}$ with VC-dimension $D(\alpha)$ under margin α, there exists a learning-to-rent algorithm that has a competitive ratio for n samples with constant failure probability (over the samples) of $1 + O(L\alpha)$, where α satisfies:*

$$\sqrt{\frac{D(\alpha)}{n}} = L\alpha. \tag{4}$$

Furthermore, the algorithm enjoys a robustness bound of $1 + \left(\frac{1}{L\alpha}\right)$

Proof. Let q denote the probability that a test/training sample (x_i, y_i) satisfies $1 - \gamma \leq y_i \leq 1 + \gamma$, i.e., is in the margin. With probability $1 - q$, a test input does not lie in the margin and we have the following two scenarios:

- With probability $(1 - \epsilon)$, the prediction is correct and the competitive ratio is at most $(1 + \gamma)$.
- With probability ϵ, the prediction is incorrect and the competitive ratio is at most $\max\left(1 + \frac{1}{\gamma}, 2 + \gamma\right)$. For small γ (say $\gamma \leq 1/2$, which will hold for any reasonable sample size n), this value is $1 + \frac{1}{\gamma}$.

With probability q, a test input lies in the margin and the competitive ratio is at most $\frac{1+\gamma}{1-\gamma}$.

Hence, the expected competitive ratio is:

$$\begin{aligned}
\textsc{cr}(\theta, \mathbb{K}) &\leq (1-q)\cdot(1-\epsilon)\cdot(1+\gamma) + \\
&\quad + (1-q)\cdot\epsilon\cdot\left(1 + \frac{1}{\gamma}\right) + q\cdot\left(\frac{1+\gamma}{1-\gamma}\right) \\
&\leq 1 + \left[(1-q)\cdot(1-\epsilon)\cdot\gamma + (1-q)\cdot\epsilon\cdot\frac{1}{\gamma} + q\cdot\frac{2\gamma}{1-\gamma}\right] \\
&\leq 1 + 4\gamma + (1-q)\cdot\frac{\epsilon}{\gamma} \qquad \text{for } \gamma \leq 1/2.
\end{aligned}$$

Now, note that by Chernoff bounds (see, e.g., Motwani & Raghavan (1997)), the number of samples used for training the classifier after filtering is $n_f \geq n(1-q)/2$ with constant probability. Also, by Theorem 6 and Lemma 7, we predict whether $y < 1$ or $y \geq 1$ with an error rate of $\epsilon = O\left(\frac{D(\alpha)}{n_f}\right)$ using n_f samples with constant probability. This implies:

$$(1-q)\cdot\epsilon = O\left(\frac{D(\alpha)}{n}\right).$$

Thus, $\textsc{cr}(\theta, \mathbb{K}) \leq 1 + 4\gamma + O\left(\frac{D(\alpha)}{n\cdot\gamma}\right)$. Optimizing for γ, we have $\gamma = \theta\left(\sqrt{\frac{D(\alpha)}{n}}\right)$. But, we also have $\gamma = L\alpha$ in the algorithm. This implies that we choose α to satisfy Eq. (4) and obtain a competitive ratio of $1 + O(L\alpha)$. For the robustness bound, note that by Lemma 1, the minimum θ set by the algorithm is γ, which yields a robustness bound of $1 + \frac{1}{\gamma} = 1 + \left(\frac{1}{L\alpha}\right)$ ☐

We now apply this theorem for the important and widely used case of linear separators. The following well-known theorem establishes the VC-dimension of linear separators with a margin.

Theorem 9 (see, e.g., Vapnik & Vapnik (1998))**.** *For an input parameter space $X \in \mathbb{R}^d$ that lies inside a sphere of radius R, the concept class of α-margin separating hyperplanes for X has the VC dimension D given by:*

$$D \leq \min\left(\frac{R^2}{\alpha^2}, d\right) + 1.$$

Feature vectors are typically assumed to be normalized to have constant norm, i.e., $R = O(1)$. Thus, Theorem 8 gives the sampling complexity for linear separators as follows:

Corollary 10. *For the class of linear separators, there is a learning-to-rent algorithm that takes as input n samples and has a competitive ratio of $1 + O\left(\frac{\sqrt{L}}{n^{1/4}}\right)$.*

For instances where a linear separator does not exist, a popular technique, called *kernelization* (see Rasmussen (2003)), is to transform the data points x to a different space $\phi(x)$ where they become linearly separable. For normalized x, if ϕ satisfies $\|\phi(x_1) - \phi(x_2)\| \geq \frac{1}{\nu}\cdot\|x_1 - x_2\|$ (we call this ν-bounded contraction), then $\|y_1 - y_2\| \leq L\nu\,\|\phi(x_1) - \phi(x_2)\|$. In other words, the Lipschitz property of the joint distribution $\mathbb{K}$ can be composed with the bounded contraction property of the kernel.

Indeed, popular kernels such as the polynomial kernel or the Gaussian kernel have the bounded contraction property. For instance, for the degree-2 polynomial kernel given by

$$\phi(x) = (1, x, x \otimes x)$$

satisfies ν-bounded contraction for $\nu = 1$.

Corollary 11. *For a kernel function ϕ satisfying $\|\phi(x_1) - \phi(x_2)\| \geq \frac{1}{\nu} \cdot \|x_1 - x_2\|$ for all x_1, x_2, assuming the data is linearly separable in kernel space, there exists a learning-to-rent algorithm that achieves a competitive ratio of $1 + O\left(\frac{\sqrt{L\nu}}{n^{1/4}}\right)$ with n samples,*

Conceptually, the corollary states that we can make use of these kernel mappings without hurting the competitive ratio bounds achieved by the algorithm. This is because the sample complexity in the margin-based algorithm (Algorithm 4) is independent of the number of dimensions.

5. Learning-to-rent with a Noisy Classifier

So far, we have seen that PAC-learning a binary classifier with deterministic labels (Assumption 1) is sufficient for a learning-to-rent algorithm. However, in practice, the data is often noisy, which leads us to relax Assumption 1 in this section. Instead of requiring $y|x$ to be deterministic, we only insist that $y|x$ is predictable with sufficient probability. In other words, we replace Assumption 1 with the following (weaker) assumption:

Assumption 1'. *In the input distribution $(x, y) \sim \mathbb{K}$, there exists a deterministic function f and a parameter p such that the conditional distribution of $y|x$ satisfies $y = f(x)$ with probability at least $1 - p$.*

This definition follows the setting of binary classification with noise first introduced by Bylander (1994). Indeed, there are several noise-tolerant binary classifiers in the literature (e.g., (Blum et al., 1998; Awasthi et al., 2014; Natarajan et al., 2013)), which leads us to ask if these classifiers can be utilized to design learning-to-rent algorithms under Assumption 1'.

We answer this question in the affirmative by designing a learning-to-rent algorithm in this noisy setting (see Algorithm 5). This algorithm assumes the existence of a binary classifier than can tolerate a noise rate of p and achieves classification error of ϵ. Let $p_0 = \max(p, \epsilon)$. If p_0 is large, then the noise/error rate is too high for the classifier to give reliable information about test data; in this case, the algorithm reverts to a worst-case (randomized) strategy. On the other hand, if p_0 is small, the the algorithm uses the label output by the classifier, but with a minimum wait time of $\sqrt{p_0}$ on all instances to make it robust to noise and/or classification error.

The next theorem shows that this algorithm has a competitive ratio of $1+O(\sqrt{p_0})$ for small p_0, and does no worse than the worst case bound of $\frac{e}{e-1}$ irrespective of the noise/error. The proof of the theorem is deferred to the appendix.

Algorithm 5 Learning-to-rent with a noisy classifier

```
Learning: Set $p_0 = \max(p, \epsilon)$.
if $p_0 \leq \frac{1}{9(e-1)^2}$ then PAC-learn the classifier on $n$ (noisy) training samples.
For test input $x$:
if $p_0 > \frac{1}{9(e-1)^2}$
then $\mathbb{P}[\theta(x) = z] = \begin{cases} \frac{e^z}{e-1}, & \text{for } z \in [0,1] \\ 0, & \text{for } z > 1. \end{cases}$
else
    if PAC-learner predicts $y < 1$
    then $\theta(x) = 1$
    else $\theta(x) = \sqrt{p_0}$.
```

Theorem 12. *If there is a PAC-learning algorithm that can tolerate noise of p and achieve accuracy ϵ, the above algorithm achieves a competitive ratio of $\min(1 + 3\sqrt{p_0}, \frac{e}{e-1})$ where $p_0 = \max\{p, \epsilon\}$.*

We also show that the above result is optimal in a rather strong sense: namely, even with no classification error, the competitive ratio achieved cannot be improved. The proof of this theorem is also deferred to the appendix.

Theorem 13. *For a given noise rate $p \leq \frac{1}{2}$, no (randomized) algorithm can achieve a competitive ratio smaller than $1 + \frac{\sqrt{p}}{2}$, even when the algorithm has access to a PAC-learner that has zero classification error.*

6. Numerical Simulations

In this section, we use numerical simulations to evaluate the algorithms that we designed for the learning-to-rent problem: the black box algorithm (Algorithm 3), the margin-based algorithm (Algorithm 4), and the algorithm for a noisy classifier (Algorithm 5). We compare the first two algorithms and show that as the predicted by the theoretical analysis, the margin-based algorithm substantially outperforms the black box algorithm in high dimensions. For learning-to-rent with a noisy classifier, we show that its competitive ratio follows the $(1 + \sqrt{p})$-curve predicted by the theoretical analysis with increasing noise rate p.

Experimental Setup. We first describe the joint distribution $(x, y) \sim \mathbb{K}$ used in the experiments. We choose a random vector $W \in \mathbb{R}^d$ whose coordinates are standard Gaussians. We view W as a hyper-plane passing through the origin ($W^T x = 0$). The value of y, representing the length of the ski season, is calculated as $\frac{2}{(1+e^{-W^T x})}$, such that $y \geq 1$ when $W^T x \geq 0$ and $y < 1$ otherwise. Note that this satisfies the Lipschitz condition given in Definition 4, with $L = 2$ for $\|W\| \leq 1$.

The input x is drawn from a mixture distribution, where with probability $1/2$ we sample x from a Gaussian $x \sim N(0, \frac{1}{d} I)$, and with probability $1/2$, we sample x as $x =$

$\alpha W + \eta$, here $\alpha \sim N(0,1)$ is a coefficient in direction W and $\eta \sim N(0, \frac{1}{d}I)$. Choosing x from the Gaussian distribution ensures that the data-set has no margin; however, in high dimensions, $W^T x$ will concentrate in a small region, which makes all the label y very close to 1. We address this issue by mixing in the second component which ensures that the distribution of y is diverse.

We test our algorithms for number of dimensions $d = 2, 100$, and 5000. For each d, we create a large corpus of samples and select N of them randomly and designate this as the training set; the remaining samples form the test set. For a given training set, we split it in two equal halves, the first half is used to train our PAC learner and the second half is used as a validation set to optimize the design parameters in the algorithms, namely τ in Algorithm 3 and γ in Algorithm 4. (We give details of this parameter optimization procedure in the appendix.) We evaluate the performance of our algorithms as N increases.

Comparison between the two algorithms. The comparative performance of Algorithm 3 and Algorithm 4 for $d = 2, 100$, and 5000 is given in Fig. 1.[3] For small d ($d = 2$), we do not see a significant difference in the performance of the two algorithms because the curse of dimensionality suffered by Algorithm 3 is not prominent at this stage. In fact, in this case the optimal margin on validation set is very close to 0. However, as d increases, Algorithm 4 starts outperforming Algorithm 3 as expected from the theoretical analysis. For $d = 100$, this difference of performance is prominent at small sample size but disappears for larger samples, because of the trade-off between sample size and number of dimensions in Corollary 10 and Theorem 4. Eventually, at $d = 5000$, Algorithm 4 is clearly superior.

To further understand the difference between the black box approach and the margin-based approach, in Figure 2, we plot the error of the two binary classifiers used in Algorithm 3 and Algorithm 4 when $d = 5000$. Although both classifiers achieve very low accuracy on the entire data-set, the margin-based classifier was able to correctly label the data points that are far from the decision boundary, i.e., the data points where mis-classification would be costly from the optimization perspective. As a result, Algorithm 4 performs much better overall.

Learning with noise. We now evaluate the learning-to-rent algorithm with a noisy classifier (Algorithm 5), We fix the number of dimensions $d = 100$, and create a training set of $N = 10^5$ samples using the same distribution as earlier. But now, we add noise to the data by declaring each data point as noisy with probability p (we will vary the parameter p over our experiments). There are two types of noisy data points: ones where the classifier predicts $y \geq 1$ and the actual value is $y < 1$, or vice-versa. For data points of the first type, we choose y from the worst case input distribution for the ski rental problem, which turns out to be the following: $\mathbb{P}[y = z] = \frac{e}{e-1} \cdot z \cdot e^{-z}$ for $z \in [0,1]$ and point mass of $1/(e-1)$ at some $z > 1$, say at $z = 2$. For data points of the second type, the input distribution is not crucial, so we simply choose a uniform random y in $[1, 2]$. The testing is done on a batch of 1000 samples from the same distribution. We use a noise tolerant Perceptron Learner (see, e.g., Bylander (1994)) to learn the classes ($y \geq 1$ and $y < 1$) in the presence of noise. We can see that even for noise rates as high as 40%, the competitive ratio of the learning-to-rent algorithm is still better than the $\frac{e}{e-1}$ that is the best achievable in the worst case. (Figure 3)

7. Conclusion and Future Work

In this paper, we explored the question of customizing learning algorithms for optimization tasks, by incorporating optimization objectives in the loss function. We demonstrated, using PAC learning, that for the classical rent or buy problem, the sample complexity of learning can be substantially improved by incorporating the insensitivity of the objective to mis-classification near the classification boundary. In addition, even if the test data is adversarial, rather than being drawn from the same distribution as the training data, then these techniques still enjoy some worst-case bounds for robustness. This general approach of "learning for optimization" opens up a new direction for future research at the boundary of machine learning and algorithm design, by providing an alternative "white box" approach to the existing "black box" approaches for using ML predictions in *beyond worst case* algorithm design. While we explored this for an online problem in this paper, the principle itself can be applied to any scenario where an algorithm hopes to learn patterns in the input that can be exploited to achieve performance gains. We posit that this is a rich direction for future research.

---

[3] In all the figures, the vertical bars represent standard deviation of the output value and the value plotted on the curve is the mean.

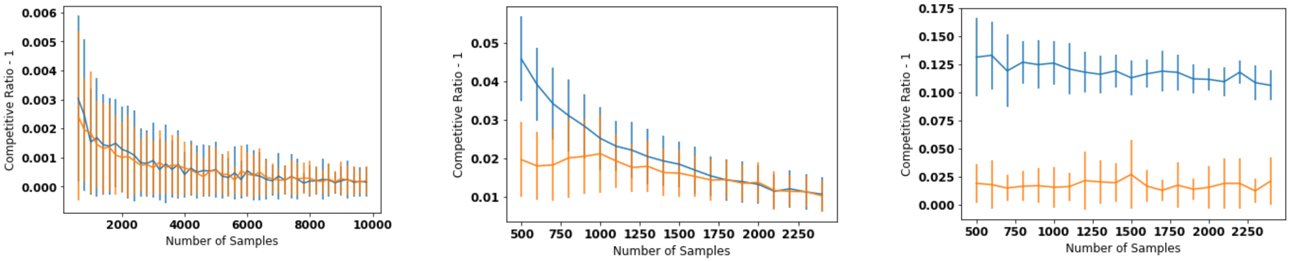

Figure 1. Comparison of Algorithm 3 (blue) and Algorithm 4 (orange). From left to right, $d = 2, 100$, and 5000.

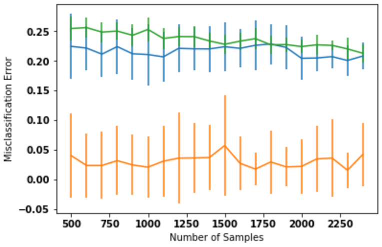

Figure 2. Classification error in Algorithm 3 (green) and Algorithm 4 (blue for all samples, orange for filtered samples).

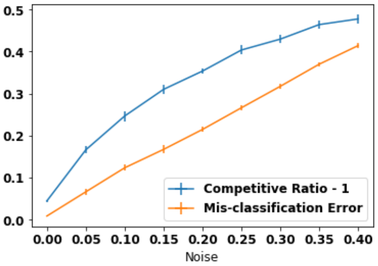

Figure 3. Algorithm 5 with varying noise rate with $d = 100$.

Acknowledgments

K. Anand and D. Panigrahi were supported in part by NSF grants NSF CCF-1527084, CCF-1535972, CCF-1955703, an NSF CAREER Award CCF-1750140, and an Indo-US Virtual Networked Joint Center on Algorithms under Uncertainty. R. Ge were supported in part by NSF CCF-1704656, NSF CCF-1845171 (CAREER), a Sloan Fellowship and a Google Faculty Research Award.

References

Ailon, N., Chazelle, B., Clarkson, K. L., Liu, D., Mulzer, W., and Seshadhri, C. Self-improving algorithms. *SIAM Journal on Computing*, 40(2):350–375, 2011.

Awasthi, P., Balcan, M. F., and Long, P. M. The power of localization for efficiently learning linear separators with noise. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pp. 449–458. ACM, 2014.

Balkanski, E., Rubinstein, A., and Singer, Y. The power of optimization from samples. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 4017–4025, 2016.

Balkanski, E., Rubinstein, A., and Singer, Y. The limitations of optimization from samples. In Hatami, H., McKenzie, P., and King, V. (eds.), *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pp. 1016–1027. ACM, 2017.

Blum, A., Frieze, A., Kannan, R., and Vempala, S. A polynomial-time algorithm for learning noisy linear threshold functions. *Algorithmica*, 22(1-2):35–52, 1998.

Bylander, T. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the seventh annual conference on Computational learning theory*, pp. 340–347, 1994.

Cole, R. and Roughgarden, T. The sample complexity of revenue maximization. In Shmoys, D. B. (ed.), *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pp. 243–252. ACM, 2014.

Elkan, C. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pp. 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Gollapudi, S. and Panigrahi, D. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pp. 2319–2327, 2019.

Gupta, R. and Roughgarden, T. A pac approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, pp. 13–30, 1963.

Hsu, C.-Y., Indyk, P., Katabi, D., and Vakilian, A. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1lohoCqY7`.

Huang, C., Zhai, S., Talbott, W., Bautista, M. A., Sun, S.-Y., Guestrin, C., and Susskind, J. Addressing the loss-metric mismatch with adaptive loss alignment. *arXiv preprint arXiv:1905.05895*, 2019.

Jiang, Z., Panigrahi, D., and Sun, K. Online algorithms for weighted caching with predictions. In *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020*, 2020.

Kamalaruban, P. and Williamson, R. C. Minimax lower bounds for cost sensitive classification. *arXiv preprint arXiv:1805.07723*, 2018.

Karlin, A. R., Manasse, M. S., Rudolph, L., and Sleator, D. D. Competitive snoopy caching. *Algorithmica*, 3: 77–119, 1988.

Karlin, A. R., Manasse, M. S., McGeoch, L. A., and Owicki, S. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.

Karlin, A. R., Kenyon, C., and Randall, D. Dynamic TCP acknowledgment and other stories about e/(e-1). *Algorithmica*, 36(3):209–224, 2003.

Kearns, M. J., Vazirani, U. V., and Vazirani, U. *An introduction to computational learning theory*. MIT press, 1994.

Khanafer, A., Kodialam, M., and Puttaswamy, K. P. N. The constrained ski-rental problem and its application to online cloud cost optimization. In *Proceedings of the INFOCOM*, pp. 1492–1500, 2013.

Kodialam, R. Competitive algorithms for an online rent or buy problem with variable demand. *SIAM Undergraduate Research Online*, 7:233–245, 2014.

Lattanzi, S., Lavastida, T., Moseley, B., and Vassilvitskii, S. Online scheduling via learned weights. In Chawla, S. (ed.), *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pp. 1859–1877. SIAM, 2020.

Ling, C. X. and Sheng, V. S. Cost-sensitive learning and the class imbalance problem, 2008.

Lotker, Z., Patt-Shamir, B., and Rawitz, D. Rent, lease or buy: Randomized algorithms for multislope ski rental. In *Proceedings of the 25th Annual Symposium on the Theoretical Aspects of Computer Science (STACS)*, pp. 503–514, 2008.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. *arXiv preprint arXiv:1802.05399*, 2018.

Medina, A. M. and Vassilvitskii, S. Revenue optimization with approximate bid predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pp. 1858–1866, 2017.

Meyerson, A. The parking permit problem. In *Proc. of 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 274–284, 2005.

Mitzenmacher, M. A model for learned bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems*, pp. 464–473, 2018.

Mitzenmacher, M. Scheduling with predictions and the price of misprediction. In Vidick, T. (ed.), *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pp. 14:1–14:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Morgenstern, J. and Roughgarden, T. Learning simple auctions. In Feldman, V., Rakhlin, A., and Shamir, O. (eds.), *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pp. 1298–1318. JMLR.org, 2016.

Motwani, R. and Raghavan, P. *Randomized Algorithms*. Cambridge University Press, 1997.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In *Advances in neural information processing systems*, pp. 1196–1204, 2013.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ml predictions. In *Advances in Neural Information Processing Systems*, pp. 9661–9670, 2018.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pp. 63–71. Springer, 2003.

Rohatgi, D. Near-optimal bounds for online caching with machine learned advice. In Chawla, S. (ed.), *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pp. 1834–1845. SIAM, 2020.

Vapnik, V. and Vapnik, V. Statistical learning theory wiley. *New York*, pp. 156–160, 1998.

A. Omitted Details for Section 3: A General Learning-to-Rent Algorithm

In this section we give the proofs for Theorem 2 and Theorem 3.

A.1. Proof of Theorem 2

Before we design an algorithm for the more general case of a Lipschitz joint distribution, let us first consider a warm-up example where we assume that x is a constant and therefore, we just have $y \sim \mathbb{K}$. We repeat the algorithm as given before in Algorithm 1, and we prove the following guarantee:

Algorithm 6 Outputs θ_A for a given distribution on y

```
Query $\left(\frac{\delta}{\epsilon^6}\right)$ samples. Initialize array $l$ of length $\frac{1}{\epsilon^2}$
Let $\ell[\theta] \leftarrow$ average of $g(\theta, y)$ over all samples $y$.
return $\theta_A \leftarrow \arg\min_{\theta \in [\epsilon, 1/\epsilon], \theta/\epsilon \in \mathbb{N}} \ell[\theta]$
```

Theorem 14. *If a learning-to-rent problem has only one possible input x, then there exists an algorithm requiring $O\left(\frac{\delta}{\epsilon^6}\right)$ samples that achieves $(1+\epsilon)$ accuracy with probability $\geq 1 - O\left(\frac{e^{-\Omega\left(\frac{\delta}{\epsilon}\right)}}{\epsilon^2}\right)$, for $0 < \epsilon < 0.1$.*

To prove this theorem, let $\mathbb{K}$ be the distribution of y, and θ^* be the optimal threshold for this distribution and $f^*_{\mathbb{K}}$ is the optimal expected competitive ratio. We first show that it suffices to get a threshold that is not much larger than θ^*:

Lemma 15. *Given that the ski-renting season $y \sim \mathbb{K}$. For $0 < \epsilon < 0.5$:*

$$\textsc{cr}(\theta^* + \epsilon, \mathbb{K}) \leq (1+\epsilon) f^*_{\mathbb{K}}$$

where $f^*_{\mathbb{K}} = \textsc{cr}(\theta^*, \mathbb{K})$ is the optimal value and the optimal threshold $\theta^* = \arg\min_{\theta \in \mathbb{R}^+} \textsc{cr}(\theta, \mathbb{K})$.

Proof. We compare the competitive ratio at different values of y. Recall that :

$$g(\theta, y) = \begin{cases} \frac{(1+\theta)}{\min\{1,y\}} & \text{if } y \geq \theta \\ \frac{y}{\min\{1,y\}} & \text{otherwise} \end{cases}$$

When $y \leq \theta^*$ then both thresholds will lead to the same cost and $g(., y)$ remains unchanged.

For $\theta^* + \epsilon > y > \theta^*$ we have

$$\frac{g(\theta^* + \epsilon, y)}{g(\theta^*, y)} = \frac{y}{1+\theta^*} \leq 1.$$

Finally, for $y > \theta^* + \epsilon$, we have

$$\frac{g(\theta^* + \epsilon, y)}{g(\theta^*, y)} = \frac{1+\theta^*+\epsilon}{1+\theta^*} \leq (1+\epsilon).$$

Since the ratio is bounded above by $1+\epsilon$ for all y, after taking the expectation we have

$$\mathbb{E}_{y\sim\mathbb{K}}[g(\theta^* + \epsilon, y)] \leq (1+\epsilon)\mathbb{E}_{y\sim\mathbb{K}}[g(\theta^*, y)].$$

□

The next lemma shows that without loss of generality we only need to consider thresholds between ϵ and $1/\epsilon$:

Lemma 16. *Let $f^\epsilon_{\mathbb{K}} = \min_{\theta \in \left[\epsilon, \frac{1}{\epsilon}\right]} \textsc{cr}(\theta, \mathbb{K})$ and $\epsilon \in [0, 0.05)$, then:*

$$f^\epsilon_{\mathbb{K}} \leq f^*_{\mathbb{K}}(1+\epsilon),$$

where $f^*_{\mathbb{K}} = \textsc{cr}(\theta^*, \mathbb{K})$ is the optimal value and the optimal threshold $\theta^{=} \arg\min_{\theta \in \mathbb{R}^+} \textsc{cr}(\theta, \mathbb{K})$.

Proof. Let $\theta^\epsilon = \arg\min_{\theta \in \left[\epsilon, \frac{1}{\epsilon}\right]}$ be the optimal threshold within the range $[\epsilon, 1/\epsilon]$. We consider different cases for the optimal threshold (without constraints) θ^*.

Case I: When $\theta^* \in \left[\epsilon, \frac{1}{\epsilon}\right]$ then clearly we have $\theta^* = \theta^\epsilon$.

Case II : $\theta^* < \epsilon$, in this case we show that choosing $\theta = \theta^* + \epsilon$ is good enough: by Lemma 15, we have that $\theta^* + \epsilon \in \left[\epsilon, \frac{1}{\epsilon}\right]$ and, $f^\epsilon_{\mathbb{K}} \leq \text{CR}(\theta^* + \epsilon, \mathbb{K}) \leq (1+\epsilon) f^*_{\mathbb{K}}$.

Case III : $\theta^* > \frac{1}{\epsilon}$, in this case we show that choosing $\theta = 1/\epsilon$ is good enough. When $y \leq 1/\epsilon$, then $g(1/\epsilon, y) \leq g(\theta^*, y)$. When $y > 1/\epsilon$, then $\frac{g(1/\epsilon, y)}{g(\theta^*, y)} \leq \frac{1/\epsilon + 1}{y} \leq \frac{1/\epsilon+1}{1/\epsilon} = 1 + \epsilon$.

Hence, $f^\epsilon_{\mathbb{K}} \leq \mathbb{E}_{y \sim \mathbb{K}}\left[g(1/\epsilon, y)\right] \leq (1+\epsilon) f^*_{\mathbb{K}}$. □

Next we show how to estimate the expected competitive ratio:

Lemma 17. *Given a fixed $\theta \in \left[\epsilon, \frac{1}{\epsilon}\right]$, by taking $\frac{\delta}{\epsilon^4}$ samples of $y \sim \mathbb{K}$, the quantity $\mathbb{E}_{y\sim\mathbb{K}}[g(\theta, y)]$ can be estimated to a multiplicative accuracy of ϵ with probability $1 - e^{-\frac{2\delta}{\epsilon}}$.*

Proof. Note that when $\theta \in \left[\epsilon, \frac{1}{\epsilon}\right]$ then $g(\theta, y)$ is bounded above by $\frac{1}{\epsilon} + 1$, therefore the random variable $g(\theta, y)$ has a variance σ^2 bounded above by $\frac{1}{\epsilon^2}$.

Let $\text{CR}(\theta, \mathbb{K}) = \mathbb{E}_{y\sim\mathbb{K}}[g(\theta, y)]$ be the true mean of the distribution and $\widehat{\text{CR}}(\theta, \mathbb{K})$ denotes the estimate that we have obtained by taking $\frac{\delta}{\epsilon^4}$ samples. Also, any estimate of $g(\theta, y)$ is from a distribution whose mean is $\text{CR}(\theta, \mathbb{K})$ and is bounded inside the range $[1, 1 + \frac{1}{\epsilon}]$. Therefore, taking $\frac{\delta}{\epsilon^4}$ samples and by Hoeffding's Inequality (Hoeffding, 1963), we claim that :

$$\mathbb{P}\left[\widehat{\text{CR}}(\theta, \mathbb{K}) - \text{CR}(\theta, \mathbb{K}) > t\right] \leq exp\left(-\frac{2\delta t}{\epsilon^2}\right).$$

Setting $t = \epsilon$ and using the fact that $\text{CR}(\theta, \mathbb{K}) \geq 1$, we get that with probability: $1 - e^{-\frac{2\delta}{\epsilon}}$,

$$\widehat{\text{CR}}(\theta, \mathbb{K}) \leq (1+\epsilon)\text{CR}(\theta, \mathbb{K}).$$

□

Now we are ready to prove Theorem 14:

Proof of Theorem 14. The algorithm simply involves dividing the segment $\left[\epsilon, \frac{1}{\epsilon}\right]$ into small intervals of ϵ width. This would give us at most $1/\epsilon^2$ intervals.(refer to Algorithm 1) For each interval $[\theta_0 - \epsilon, \theta_0]$ we use the $\frac{\delta}{\epsilon^6}$ samples at $\theta = \theta_0$ to calculate $\widehat{\text{CR}}(\theta_0, \mathbb{K})$. We output the θ_0 that has the minimum $\widehat{\text{CR}}(\theta_0, \mathbb{K})$ over all such intervals.

By Lemma 17 we know that our estimate is within a $(1+\epsilon)$ multiplicative factor of the true $\text{CR}(\theta_0, \mathbb{K})$ with probability: $1 - e^{-\frac{2\delta}{\epsilon}}$. Since there are at most $\frac{1}{\epsilon^2}$ such θ_0: by a simple union bound, we claim that all our estimates on the competitive ratio are $(1+\epsilon)$ multiplicative factor of the true $\text{CR}(\theta_0, \mathbb{K})$ with probability : $1 - \left(\frac{e^{-\frac{2\delta}{\epsilon}}}{\epsilon^2}\right)$. Also lemma 15 tells us that $\text{CR}(\theta_0, \mathbb{K})$ is within a $(1+\epsilon)$ factor of $\text{CR}(\theta, \mathbb{K})$ for all $\theta \in [\theta_0 - \epsilon, \theta_0]$. Therefore, by taking the minimum over all θ_0 : we are within a $(1 + 2\epsilon + \epsilon^2)$ factor of $\min_{\theta \in \left[\epsilon, \frac{1}{\epsilon}\right]} \text{CR}(\theta, \mathbb{K})$. Finally, we invoke lemma 16 to claim that our value is within a $(1 + 4\epsilon)$ (for $\epsilon < 0.4$) multiplicative factor of f^*. Repeating the above analysis with $\epsilon' = \frac{\epsilon}{4}$, we achieve $(1 + \epsilon')$ accuracy using $\frac{256\delta}{\epsilon'^4}$ samples with probability: $1 - 16\left(\frac{e^{-8\delta/\epsilon'}}{\epsilon'^2}\right)$. □

Using the algorithm for a single point x, we will now design the algorithm for learning-to-rent for Lipschitz distributions. As we discussed before, the idea is to discretize the space. In order for this approach to work, we need to show that if two distributions are close (in earth mover distance) then the optimal solutions for the two distributions are also close:

Lemma 18. *Given two distributions $\mathbb{D}_1, \mathbb{D}_2$ such that $EMD(\mathbb{D}_1, \mathbb{D}_2) \leq \Delta$, then:*

$$\mathbb{E}_{y\sim\mathbb{D}_1}[g(\theta + \epsilon, y)] \leq (1+\epsilon)\left(1 + \frac{\Delta}{\epsilon^2}\right)\mathbb{E}_{y\sim\mathbb{D}_2}[g(\theta, y)], \textit{ for any } \theta \in \mathbb{R}^+.$$

Proof. Let $p_i(y_0)$ be the probability that $y = y_0$ for distribution $\mathbb{D}_i$. For $y \leq \theta + \epsilon$: $\frac{g(\theta+\epsilon,y)}{g(\theta,y)} \leq 1$. Also, when $y > \theta + \epsilon$ then, $\frac{g(\theta+\epsilon,y)}{g(\theta,y)} = \frac{1+\theta+\epsilon}{1+\theta} \leq (1+\epsilon)$.

Let us begin at distribution $\mathbb{D}_2$, and there be an adversary who wants to increase the expectation $\mathbb{E}_y[g(\theta+\epsilon, y)]$ by shifting some probability mass and thereby changing the distribution. However the adversary cannot change the distribution drastically (which is where the EMD comes into play), the total earth mover distance between the new and old distribution can be at most Δ.

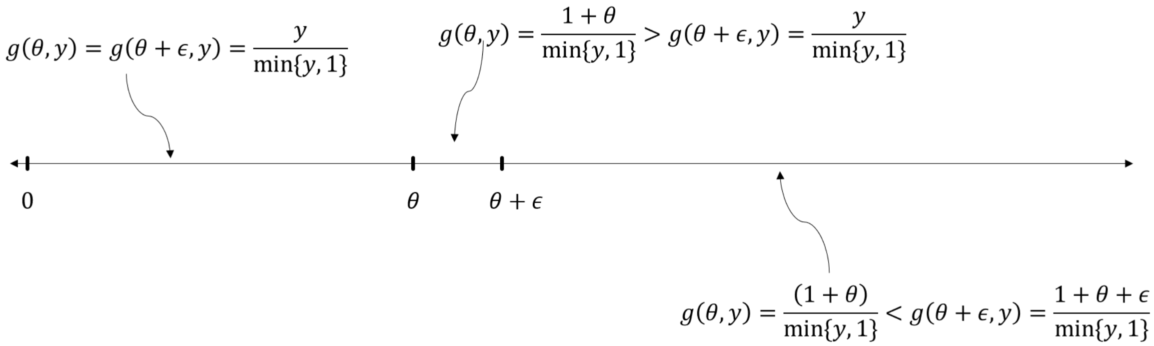

Figure 4. Value of $g(\theta, y)$ in different regimes of y

The figure 4 shows the different values of $g(\theta, y)$ and $g(\theta+\epsilon, y)$ in the regions where $y \leq \theta$, $y \in (\theta, \theta+\epsilon]$ and $y > \epsilon + \theta$. Note that the difference $g(\theta+\epsilon, y) - g(\theta, y)$ is greatest when $y > \epsilon + \theta$. Shifting any probability mass within the regions $y < \theta$ or $y > \theta + \epsilon$ does not increase the quantity $\frac{g(\theta+\epsilon,y)}{g(\theta,y)}$. If we shift some probability mass from $y_1 \in [\theta, \theta+\epsilon]$ to $y_2 > \theta + \epsilon$, the increase in $\frac{g(\theta+\epsilon,y_2)}{g(\theta,y_1)}$ is upper bounded by $1 + \epsilon$. Note that we can shift as much mass as we want from $y_1 = \theta + \epsilon - \tau$ to $y_2 = \theta + \epsilon + \tau$ for $\tau \to 0$.

The maximum change occurs when we move from $y_1 < \theta$ to $y_2 > \theta + \epsilon$, then $\frac{g(\theta+\epsilon,y_2)}{g(\theta,y_1)} = \left(\frac{\min\{1,y_1\}}{y_1} \cdot \frac{(1+\theta+\epsilon)}{\min\{1,y_2\}}\right) \leq \left(\frac{(1+\theta+\epsilon)}{\min\{1,y_2\}}\right)$. However, the maximum probability mass that can be moved is upper bounded by $\frac{\Delta}{y_2-y_1}$ (Since we know that $\mathbb{D}_1$ and $\mathbb{D}_2$ differ by Δ).

Thus the upper bound we obtain is,

$$\begin{aligned}
\frac{\mathbb{E}_{y\sim\mathbb{D}_1}[g(\theta+\epsilon, y)]}{\mathbb{E}_{y\sim\mathbb{D}_2}[g(\theta, y)]} &\leq (1+\epsilon) + \max_{y_1\in[0,\theta), y_2>\theta+\epsilon} \left(\frac{(1+\theta+\epsilon)}{\min\{1, y_2\}} \times \frac{\Delta}{y_2 - y_1}\right) \\
&= (1+\epsilon) + \left(\frac{1+\theta+\epsilon}{\theta+\epsilon} \times \frac{\Delta}{\epsilon}\right) \\
&\leq (1+\epsilon) + (1+\epsilon)\frac{\Delta}{\epsilon^2} \\
&= (1+\epsilon)\left(1 + \frac{\Delta}{\epsilon^2}\right)
\end{aligned}$$

□

As a corollary, by linearity of expectation we know if many distributions are close, then their optimal solutions are also close:

Corollary 19. *Let $\mathbb{K}_1, \mathbb{K}_2$ be two joint distributions on (X, Y) such that they have the same support S on X. If $\forall x_i, x_j \in S$, $\mathbb{D}_i = Y \mid (X = x_i), \mathbb{D}_j = Y \mid (X = x_j)$ satisfies $EMD(\mathbb{D}_i, \mathbb{D}_j) \leq \Delta$ then:*

$$\mathbb{E}_{(x,y)\sim\mathbb{K}_1}\left[g(\theta+\epsilon, y)\right] \leq (1+\epsilon)\left(1 + \frac{\Delta}{\epsilon^2}\right)\mathbb{E}_{(x,y)\sim\mathbb{K}_2}\left[g(\theta, y)\right]$$

And,

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{K}_1}\left[g(\theta,y)\right] \le (1+\epsilon)\left(1+\frac{\Delta}{\epsilon^2}\right)\min_\theta \mathbb{E}_{(x,y)\sim\mathbb{K}_2}\left[g(\theta,y)\right]$$

We now repeat Algorithm 2 below for completeness.

Algorithm 7 Procedure that outputs $\theta_A(x)$ for multi-dimensional x

```
Divide the hyper-cube $[0,1]^d$ into sub-cubes of side length $\frac{\epsilon^3}{64L\sqrt{d}}$ each.
(Note that the number of such cubes are $N = \left(\frac{64L\sqrt{d}}{\epsilon^3}\right)^d$.)
Query $\Pi = \left(\frac{1024L\sqrt{d}}{\epsilon^6}\right)^{2d}$ samples. Set threshold $\tau = \left(\frac{64L\sqrt{d}}{\epsilon^8}\right)^d$.
Index each cube by $i$ as $1 \le i \le N$.
for each sub-cube $C_i$:
  if the number of samples from the sub-cube exceed $\tau$,
    Compute (using Algorithm 1) $\theta_i \leftarrow \arg\min_\theta \mathbb{E}_{(x,y):x\in C_i}[g(\theta,y)]$.
    For all $x \in C_i$: Output $\theta_A(x) \leftarrow \theta_i$.
  else For all $x \in C_i$: Output $\theta_A(x) \leftarrow 1$.
```

Here we are ready to prove Theorem 2 which we restate for completeness:

Theorem. *For a learning-to-rent problem, if $x \in [0,1]^d$, and the joint distribution (x,y) is L-Lipschitz, then there exists an algorithm that uses $n = \left(\frac{L\sqrt{d}}{\epsilon}\right)^{O(d)}$ samples and is $(1+\epsilon)$-accurate with high probability* [4].

Proof. Let us focus on a certain sub-cube C_i, we will break them into two cases: one where the sub-cube gets enough samples and one where the sub-cube does not get enough samples.

CASE I: Let's say that we met the threshold and got over $\frac{1}{\epsilon^{8d}}$ samples in C_i. Let $\mathbb{K}_i$ be the true conditional distribution of Y when $X = x$ lies in C_i. Clearly, when we are sampling Y where X lies inside C_i our estimate might be a from a different distribution $\hat{\mathbb{K}}_i$.

But both these distributions are from a linear combinations of conditional distributions $Y \mid (X = x)$ over $x \in C_i$. Using algorithm 1 we get a θ_i for C_i and using the result from Theorem 14 (with $\delta = \frac{1}{\epsilon^{8d-6}}$), and union bound over all the cubes, we can claim that with a very high probability: $\ge 1 - O\left(\epsilon^{\Omega(d)}\right)$, it satisfies:

$$\forall i \; \mathbb{E}_{y\sim\hat{\mathbb{K}}_i}[g(\theta_i,y)] \le \left(1+\frac{\epsilon}{3}\right)\cdot \min_\theta \mathbb{E}_{y\sim\hat{\mathbb{K}}_i}[g(\theta,y)] \tag{5}$$

Using Corollary 19 we have the following

$$\min_\theta \mathbb{E}_{y\sim\mathbb{K}_i}[g(\theta,y)] \le \left(1+\frac{\epsilon}{4}\right)\left(1+\frac{16\Delta}{\epsilon^2}\right)\min_\theta \mathbb{E}_{y\sim\hat{\mathbb{K}}_i}[g(\theta,y)] \tag{6}$$

Since for any $x, y \in C$, we have $\|x-y\|_2 \le \frac{\epsilon^3}{64L}$, therefore using the Lipchitz assumption, we have $\Delta \le \frac{\epsilon^3}{64}$. Hence,

$$\min_\theta \mathbb{E}_{y\sim\hat{\mathbb{K}}_i}[g(\theta,y)] \le \left(1+\frac{\epsilon}{3}\right)\min_\theta \mathbb{E}_{y\sim\mathbb{K}_i}[g(\theta,y)]. \tag{7}$$

Using Theorem 14, and for $\epsilon < 0.1$,

$$\mathbb{E}_{y\sim\mathbb{K}_i}[g(\theta_i,y)] \le \left(1+\frac{3\epsilon}{4}\right)\min_\theta \mathbb{E}_{y\sim\mathbb{K}_i}[g(\theta,y)]. \tag{8}$$

CASE II: When C_i does not have enough samples to meet the threshold and we set $\theta_A(x) = 1$ for all $x \in C_i$. In this case, we have that $g(\theta_A(x), y) = g(1, y) \le 2$.

---

[4] with probability exceeding $1 - \epsilon^{\Omega(d)}$

We will see now that the second case occurs with a very small probability. Let $P[x \in C_i]$ be denoted as p_i and let $\hat{p}_i$ be our empirical estimation of p_i. By Hoeffding's bound,

$$\mathbb{P}[\|p_i - \hat{p}_i\| \geq t] \leq 2e^{-2\Pi \cdot t^2}.$$

where $\Pi = \left(\frac{1024L\sqrt{d}}{\epsilon^6}\right)^{2d}$ is the number of samples we took. If we set $t = \frac{\epsilon^{4d}}{(1024L\sqrt{d})^d}$, we have: $\|p_i - \hat{p}_i\| < \frac{\epsilon^{4d}}{(1024L\sqrt{d})^d}$, with probability : $\geq 1 - 2exp(-\frac{2}{\epsilon^{4d}})$. By carrying a simple union bound over all such i, we show that the above relation holds true for all C_i with probability:

$$1 - N \cdot 2e^{-\frac{2}{\epsilon^{4d}}}.$$

Using simple inequalities like $e^{-x} < \frac{1}{x^2}$ for $x > 0$ we can show that this probability is greater than $\alpha = 1 - O(\epsilon^{\Omega(d)})$.

Let a cube be termed **good** if it has the threshold satisfied and **bad** otherwise. Also, $C(x)$ denotes the cube which contains x and n_i is the number of samples lying inside cube C_i. Let $\mathbb{V}$ denote the discrete distribution of x over the cubes. The probability $p_i = \mathbb{P}_{x \sim \mathbb{V}}[x \in C_i]$ that an x chosen from $\mathbb{V}$ will lie in C_i is estimated as $\hat{p}_i = \frac{n_i}{\Pi}$ and as shown above, with probability $\geq 1 - \alpha$: $\|p_i - \hat{p}_i\| < \frac{\epsilon^{4d}}{(1024L\sqrt{d})^d}$ We obtain:

$$\begin{aligned}
\mathbb{P}_{x \sim \mathbb{V}}[C(x) \text{ is good}] &= \sum_{C_i \text{ is good}} p_i \\
&\geq \sum_{C_i \, is \, good} \frac{n_i}{\Pi} - \sum_{C_i \text{ is good}} (\|p_i - \hat{p}_i\|) \\
&\geq \left(\frac{\sum_{\text{all cubes } C_i} n_i - \sum_{C_i \text{ is bad}} n_i}{\Pi}\right) - \frac{\epsilon^{4d}}{(1024L\sqrt{d})^d} \times N \\
&\geq 1 - \left(\frac{\sum_{C_i \text{ is bad}} n_i}{\Pi}\right) - \left(\frac{\epsilon}{16}\right)^d \\
&\geq 1 - \left(\frac{N \times \tau}{\Pi}\right) - \left(\frac{\epsilon}{16}\right)^d. \\
&\geq 1 - \left(\frac{\epsilon}{16}\right)^d - \left(\frac{\epsilon}{16}\right)^d.
\end{aligned}$$

Thus,

$$\mathbb{P}_{x \sim \mathbb{V}}[C(x) \text{ is bad}] \leq 2\left(\frac{\epsilon}{16}\right)^d \leq \frac{\epsilon}{8}.$$

Therefore, if $\theta_A(x)$ is the algorithm's output and $\theta^*(x)$ is the optimal threshold, then we get:

$$\begin{aligned}
\mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta_A(x), y)] &= \left(1 + \frac{3\epsilon}{4}\right) \sum_{C_i \text{ is good}} (\min_{\theta} \mathbb{E}_{x,y \sim \mathbb{K}_i}[g(\theta, y)] \mathbb{P}_{x \sim \mathbb{V}}[x \in C_i]) + 2 \sum_{C_i \text{ is bad}} \mathbb{P}_{x \sim \mathbb{V}}[x \in C_i]) \\
&\leq \left(1 + \frac{3\epsilon}{4}\right) \mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)] + 2\mathbb{P}_{x \sim \mathbb{V}}[C(x) \text{ is bad}] \\
&\leq \left(1 + \frac{3\epsilon}{4}\right) \mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)] + \frac{\epsilon}{4}.
\end{aligned}$$

Since $\mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)] \geq 1$, we have

$$\begin{aligned}
\mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta_A(x), y)] &\leq \left(1 + \frac{3\epsilon}{4}\right) \mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)] + \frac{\epsilon}{4} \cdot \mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)] \\
&= (1 + \epsilon) \cdot \mathbb{E}_{(x,y) \sim \mathbb{K}}[g(\theta^*(x), y)].
\end{aligned}$$

$\square$

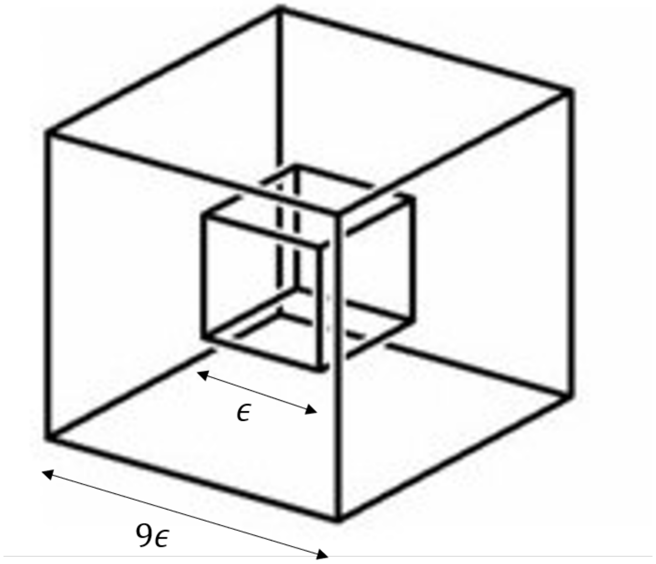

Figure 5. A sub-Hypercube with the core inside

A.2. Proof of Theorem 3

In this section, we will give a proof of Theorem 3.

In the construction, we first divide up the feature space $[0,1]^d$, which is in the form of a hypercube, into smaller hypercubes of side length 9ϵ. Note that there are $\frac{1}{(9\epsilon)^d}$ such sub-hypercubes (see fig 5). Next, we define the *core* of each sub-hypercube as a hypercube of side length ϵ at the center of the sub-hypercube. In other words, the core excludes a boundary of width 4ϵ in all dimensions. To reduce the effect of the 1-Lipschitz property, we next make two design choices. First, we define x as being uniformly distributed over the cores of all the sub-hypercubes, and the boundary regions have a probability density of 0. Second, the conditional distribution $y|x$ is deterministic and invariant in any core, with value $y = 1 - 4\epsilon$ or $y = 1 + 4\epsilon$ with probability $1/2$ each.

We now prove two key properties of this family of distributions (x, y). The first lemma shows that we have effectively eliminated the information leakage caused by the 1-Lipschitz property.

Lemma 20. *If an algorithm does not query any sample from a core, then it does not have any information about the conditional distribution $y|x$ in that core.*

Proof. Note that if $x_1, x_2 \in \mathbb{R}^d$ are in different cores, then $\|x_1 - x_2\| \geq 8\epsilon$. This implies that even with the 1-Lipschitz property, the EMD between the conditional distributions $y|x_1$ and $y|x_2$ can be 8ϵ. Since the two deterministic distributions of $y|x$ used in the construction have this EMD between them, the lemma follows. ☐

The next lemma establishes that an algorithm that does not have any information about a conditional distribution $y|x$ in any core essentially cannot do better than random guessing.

Lemma 21. *If an algorithm does not query any sample from a core, then its expected competitive ratio on the conditional distribution $y|x$ in that core is at least $1 + 2\epsilon$.*

Proof. If a rent-or-buy algorithm is specified only two possible inputs where $y = 1 - 4\epsilon$ or $y = 1 + 4\epsilon$ (for small enough $\epsilon > 0$), it has two possible strategies that dominate all others: buy at time 0 or rent throughout. The first strategy achieves a competitive ratio of $\frac{1}{1-4\epsilon} > 1 + 4\epsilon$ for $y = 1 - 4\epsilon$ and 1 or $y = 1 + 4\epsilon$, whereas the second strategy achieves a competitive ratio of 1 for $y = 1 - 4\epsilon$ and $1 + 4\epsilon$ or $y = 1 + 4\epsilon$. Since the two conditional distributions are equally likely in a core for the family of joint distributions constructed above, the lemma follows. ☐

We are now ready to prove Theorem 3 which we restate for completeness:

Theorem. *For any learning-to-rent algorithm, there exists a family of* 1*-Lipschitz joint distributions* (x, y) *where* $x \in [0,1]^d$ *such that the algorithm must query* $\frac{1}{\epsilon^{\Omega(d)}}$ *samples in order to be* $(1+\epsilon)$*-accurate, for small enough* $\epsilon > 0$*.*

Proof of Theorem 3. Assume, if possible, that the algorithm uses $n = 1/\epsilon^{d/4}$ samples. Recall that we have $1/(9\epsilon)^d > 1/\epsilon^{d/2} = n^2$ sub-hypercubes (for small enough ϵ) in the construction above. This implies that for at least $1 - 1/n$ fraction of the sub-hypercubes, the algorithm does not get any sample from them. By Lemmas 20 and 21, the competitive ratio of the algorithm on these sub-hypercubes is no better than $1 + 2\epsilon$. Therefore, even if the algorithm achieved a competitive ratio of 1 on the other sub-hypercubes. the overall competitive ratio is no better than $(1 - 1/n) \cdot (1 + 2\epsilon) + (1/n) \cdot 1 > 1 + \epsilon$. The theorem now follows from the observation that an optimal algorithm that knows the conditional distributions $y|x$ in all sub-hypercubes achieves a competitive ratio of 1. □

B. Omitted Details for Section 4: A PAC-Learning Approach to the Learning-to-Rent Problem

In this section we prove Theorems 4 and Theorem 5.

Recall Theorem 4 (restated below) shows that a PAC learning algorithm always implies a learning-to-rent algorithm.

Theorem. *Given an algorithm that PAC learns the concept class* $\mathcal{C}$ *with error* ϵ *and failure probability* δ*, there exists a learning-to-rent algorithm that has a competitive of* $(1 + 2\sqrt{\epsilon})$ *with probability* $1 - \delta$*.*

Proof. The algorithm first uses PAC-learning as a black box to learn a hypothesis $\hat{c}$. We then set $\theta(x) = 1$ when $\hat{c}(x) = 0$ and setting $\theta(x) = \tau$ (for some small τ that we fix later) when $\hat{c}(x) = 1$.

If $\mathbb{D}$ denotes the distribution of input parameter x then we know that,

$$\mathbb{P}_{x \sim D}[c(x) \neq \hat{c}(x)] \leq \epsilon. \tag{9}$$

Obviously, when $\hat{c}(x) = c(x) = 1$, then our worst-case competitive ratio is $1 + \tau$. When $\hat{c}(x) = c(x) = 0$, then our competitive ratio is 1. Also with probability ϵ, $c(x) \neq \hat{c}(x)$ and the worst case competitive ratio is $\max(2, 1 + 1/\tau)$.

If we use $\tau = \sqrt{\epsilon}$, we see that the competitive ratio CR is bounded above as:

$$\begin{aligned} \textsc{cr}(\theta, \mathbb{K}) &\leq \left(1 + \frac{1}{\tau}\right) \cdot \epsilon + (1 - \epsilon) \cdot (1 + \tau) \\ &= 1 + \frac{\epsilon}{\tau} + \tau \cdot (1 - \epsilon) \leq 1 + 2\sqrt{\epsilon}. \end{aligned}$$

Hence, with probability $1 - \delta$, we achieve a competitive ratio of $(1 + 2\sqrt{\epsilon})$. The robustness bounds follows immediately from Lemma 1 by noting that $\theta \geq \sqrt{\epsilon}$ for all inputs. □

Theorem 5 (restated below) shows that every learning-to-rent algorithm can also be converted to a PAC learning algorithm.

Theorem. *If there exists an* (ϵ, δ)*-accurate learning-to-rent algorithm for a concept class* $\mathcal{C}$ *with* n *samples, then there exists an* $O(\epsilon, \delta)$*-PAC learning algorithm for* $\mathcal{C}$ *with the same number of samples.*

Proof. We will design a PAC learning algorithm (call it P) using the learning-to-rent algorithm (call it A). Given a sample (x_i, z_i) for P, we define sample (x_i, y_i) for A where $y_i = 10$ when $z_i = 1$, and $y_i = 0$ or $y = \frac{1}{2}$ with probability $\frac{1}{2}$ each, when $z_i = 0$. The output for P for a feature x is decided as follows: when $\theta(x) \geq \frac{1}{2}$ predict $\hat{z} = 0$, otherwise, predict $\hat{z} = 1$.

First, we calculate the probability $\mathbb{P}[\hat{z} = 0, z = 1]$. When P predicts $\hat{z} = 0$, then we have $\theta(x) \geq \frac{1}{2}$. But if, $z = 1$, then the optimal cost is 1, whereas the algorithm pays at least $\frac{3}{2}$. Hence the competitive ratio is bounded below by $\frac{3}{2}$. Since the overall competitive ratio is less than $(1 + \epsilon)$, we have that:

$$(1 - \mathbb{P}[\hat{z} = 0, z = 1]) + (3/2) \cdot \mathbb{P}[\hat{z} = 0, z = 1] \leq (1 + \epsilon).$$

Therefore, $\mathbb{P}[\hat{z} = 0, z = 1] \leq 2\epsilon$.

Second, we calculate the error on the other side which is the probability $\mathbb{P}[\hat{z} = 1, z = 0]$. When P predicts $\hat{z} = 1$, then we have $\theta(x) < \frac{1}{2}$. But if $z = 0$, then $y = \frac{1}{2}$ with probability $\frac{1}{2}$, and therefore, the competitive ratio is ≥ 2

with probability $\frac{1}{2}$. With the remaining probability of $\frac{1}{2}$, we have $y = 0$, and therefore, the competitive ratio is ≥ 1. Therefore, the overall competitive ratio when $\hat{z} = 1, z = 0$ is $\geq \frac{2+1}{2} = \frac{3}{2}$. As earlier, we have $\mathbb{P}[\hat{z} = 1, z = 0] \leq 2\epsilon$. $\mathbb{P}[\hat{z} \neq z] = \mathbb{P}[\hat{z} = 1, z = 0] + \mathbb{P}[\hat{z} = 0, z = 1] \leq 4\epsilon$. □

Lastly we prove Lemma 7 which is used in the proof of Theorem 8 (in the main body).

Proof of Lemma 7. Let (x_1, y_1) and (x_2, y_2) be two samples that have been accepted such that $y_1 < 1 - \gamma$ and $y_2 > 1 + \gamma$. Clearly, $y_2 - y_1$ exceeds 2γ which means that $\|x_1 - x_2\|$ can not be less than $\frac{2\gamma}{L} = 2\alpha$. Hence, the filtered sample set has a margin of α. □

C. Algorithm for Asymmetric PAC Learning

In this section, we point out that the binary classification errors on one side are more crucial for Algorithm 3, and the following is a tighter result than Theorem 4 when the algorithm is given access to a PAC Learner with asymmetrical errors.

Definition 7. *Given a set $X \in \mathbb{R}^d$ and a concept class C of Boolean functions $X \to \{0,1\}$. Let there be an arbitrary hypothesis $c \in C$. Let P be a PAC learning algorithm that takes as input the set S comprising of m samples (x_i, y_i) where x_i is sampled from a distribution $\mathbb{D}$ on X and $y_i = c(x_i)$, and outputs a hypothesis $\hat{c}$. P is said to be have an (α, β) error with failure probability δ, if with probability at least $1 - \delta$ on the sampling of set S.*

$$\mathbb{P}_{x\sim\mathbb{D}}[\hat{c}(x) = 0, c(x) = 1] \leq \alpha$$
$$\mathbb{P}_{x\sim\mathbb{D}}[\hat{c}(x) = 1, c(x) = 0] \leq \beta$$

Theorem 22. *Given an algorithm that PAC learns the concept class C with asymmetrical errors (α, β) and failure probability δ, there exists an algorithm that has a competitive of $(1 + 3\sqrt{\epsilon})$ with probability $1 - \delta$, where $\epsilon = \max(\alpha, \sqrt{\beta})$*

Proof. Again we use PAC-learning as a black box to learn a hypothesis $\hat{c}$. We then set $\theta(x) = 1$ when $\hat{c}(x) = 0$ and setting $\theta(x) = \tau$ (for some small τ that will be decided later) when $\hat{c}(x) = 1$

Note that with probability α, $\hat{c}(x) = 0$ and $c(x) = 1$, then we have competitive ratio being capped at 2. And with probability β, $\hat{c}(x) = 1$ and $c(x) = 0$, and our competitive ratio in this case is $\frac{1+\tau}{\tau}$. The rest of the cases, we have the competitive ratio capped at $1 + \tau$.

The expected CR is therefore,

$$\begin{aligned}
\text{CR} &\leq \beta(1 + \frac{1}{\tau}) + 2\alpha + (1 - \alpha - \beta)(1 + \tau) \\
&= 1 + \alpha(1 - \tau) + \tau + \frac{\beta}{\tau} \\
&\leq 1 + \alpha + \tau + \frac{\beta}{\tau}
\end{aligned}$$

The CR is minimized at $\tau = \epsilon = \max(\alpha, \sqrt{\beta})$ and its value is $1 + 3\epsilon$. □

D. Omitted Details for Section 5: Learning-to-Rent with a Noisy Classifier

In this section we prove Theorem 12 (restated below), which gives the guarantee for learning-to-rent algorithm under noise.

Theorem. *If there is a PAC-learning algorithm that can tolerate noise of p and achieve accuracy ϵ, the above algorithm achieves a competitive ratio of $\min(1 + 3\sqrt{p_0}, \frac{e}{e-1})$ where $p_0 = \max\{p, \epsilon\}$.*

Proof of Theorem 12. Note that if $p_0 > \frac{1}{9(e-1)^2}$, we choose the threshold θ according to:

$$\Pr[\theta = z] = \begin{cases} \frac{e^z}{e-1}, & \text{for } z \in [0,1] \\ 0, & \text{for } z > 1. \end{cases}$$

It can be verified by taking the expectation that $\mathbb{E}[Alg] = \frac{e}{e-1} \times \min\{y, 1\}$ and we obtain the competitive ratio $\frac{e}{e-1}$ (Karlin et al., 1994).

Hence, we assume that $p_0 < \frac{1}{9(e-1)^2}$ for the rest of the proof.

We first focus on the points where the PAC learner's prediction is correct. This is indeed true for $1-\epsilon$ fraction of the samples from the distribution, where the expectation is taken over the probability distribution of the samples.

If $y > 1$, then the algorithm chooses to buy at $\sqrt{p_0}$, the adversary can flip the label and cause the CR (in the worst-case) to become $1 + \frac{1}{\sqrt{p_0}}$ (this happens with probability p), and otherwise, the competitive ratio is upper bounded by $(1 + \sqrt{p_0})$ (with probability $1-p$). Hence, in expectation the competitive ratio is therefore $p(1 + \frac{1}{\sqrt{p_0}}) + (1-p)(1 + \sqrt{p_0}) < 1 + \sqrt{p_0} + \frac{p}{\sqrt{p_0}}$.

When $y < 1$ and $p \le p_0 \le \frac{1}{9(e-1)^2}$, we buy at 1 and our competitive ratio is 2 with probability p (adversarial) and 1 with probability $1-p$ (no adversary). Hence, the expected competitive ratio is $1 + 2p$.

Let's focus on the points on which the PAC learner makes an error. These comprise ϵ fraction of the points in the distribution. When y is predicted to be ≤ 1 and is actually > 1, then our competitive ratio is upper bounded by 2 (since we our always buying before y exceeds 1 and the optimal solution pays 1). When y is predicted to be > 1 but is actually $y < 1$, then the worst case competitive ratio is $1 + \frac{1}{\sqrt{p_0}}$.

We are now ready to calculate the expected competitive ratio as follows:

$$\begin{aligned}\text{CR} &\le (1 + \sqrt{p_0} + \frac{p}{\sqrt{p_0}})(1-\epsilon) + \epsilon(1 + \frac{1}{\sqrt{p_0}}) \\ &\le 1 + (1-\epsilon)\sqrt{p_0} + (1-\epsilon)\frac{p}{\sqrt{p_0}} + \frac{\epsilon}{\sqrt{p_0}} \\ &\le 1 + 3\sqrt{p_0}.\end{aligned}$$

□

Now we restate Theorem 13 and prove it:

Theorem. *For a given noise rate $p \le \frac{1}{2}$, no (randomized) algorithm can achieve a competitive ratio smaller than $1 + \frac{\sqrt{p}}{2}$, even when the algorithm has access to a PAC-learner that has zero classification error.*

Proof of Theorem 13. We will show that the adversary can choose a distribution on supplying y that yields a large competitive ratio regardless of the θ that the algorithm chooses. Let's focus when $y > 1$ and the PAC learner correctly predicts this surely.

If there was no adversary, the algorithm should buy at 0 and CR is 1. However the presence of an adversary makes it a bad move, since the adversary can pick $y = \rho$ for an arbitrarily small but positive ρ with a non-zero probability and drive up the competitive ratio arbitrarily.

Here is the exact strategy that the adversary chooses to hurt the algorithm: the distribution on y is $g(y) = kye^{-y}$. for $y \in [0, \sqrt{p}]$ (k being the normalization constant) This is quite similar to the adversarial distribution chosen in (Karlin et al., 1994) to enforce an $\frac{e}{e-1}$ ratio.

Now for any value $\theta \in (0, \sqrt{p})$ that the algorithm chooses, the competitive ratio is given by:

$$\text{CR}(\theta, \mathbb{K}) = p \cdot \left[\int_0^\theta g(y)dy + \int_\theta^{\sqrt{p}} \frac{(1+\theta)}{y} g(y)dy\right] + (1+\theta)(1-p).$$

Calculating the derivative with respect to θ, we get:

$$\begin{aligned}
\frac{d(\text{CR}(\theta,\mathbb{K}))}{d(\theta)} &= pg(\theta) + p\int_{\theta}^{\sqrt{p}} \frac{g(y)}{y}dy - p\frac{(1+\theta)}{\theta}g(\theta) + (1-p) \\
&= p\frac{g(\theta)}{\theta} + p\int_{\theta}^{\sqrt{p}} ke^{-y}dy + (1-p) \\
&= -pke^{-\theta} + pk(e^{-\theta} - e^{-\sqrt{p}}) + (1-p) \\
&= -pke^{-\sqrt{p}} + (1-p)
\end{aligned}$$

Using the fact that the total probability $\int_0^{\sqrt{p}} g(y)dy = 1$ we get that $k = \frac{1}{(1-(1+\sqrt{p})e^{-\sqrt{p}})}$. It is easy to see that this value of k gives: $\frac{d(\text{CR}(\theta,\mathbb{K})}{d(\theta)} \leq 0$.

Hence $\text{CR}(\theta,\mathbb{K})$ decreases as θ goes from 0 to $\sqrt{p}$. Also, the algorithm gains nothing by increasing θ beyond $\sqrt{p}$. Hence, the best competitive ratio is obtained when algorithm chooses $\theta = \sqrt{p}$. Thus, the algorithm can't hope for a competitive ratio better than

$$\begin{aligned}
\text{CR}(\sqrt{p},\mathbb{K}) &= p\int_0^{\sqrt{p}} g(y)dy + (1+\sqrt{p})(1-p) \\
&= p + (1+\sqrt{p})(1-p) \\
&= 1 + \sqrt{p} - p\sqrt{p}
\end{aligned}$$

For $p < 1/2$:

$$\geq 1 + \frac{\sqrt{p}}{2}$$

$\square$

E. Details of Experiments

Parameter Optimization for Algorithm 3 and Algorithm 4. In this section, we give details of how we optimize the parameters used in the learning-to-rent algorithms used in our numerical simulations. As described earlier, we perform this optimization on a validation set that is distinct from the training set for these algorithms.

For the black box algorithm (Algorithm 3), we have to choose the value of the parameter τ. Here we set $\tau = c\epsilon$ where $c > 0$ is a parameter that we optimize on the validation set. In order to do this, we minimize the loss (in this case, the competitive ratio) by running gradient descent from a starting value c_0, where $c_0 \in \{1000, 100, 10, 1, 0.1, 0.01\}$.

For the margin based learning-to-rent algorithm (Algorithm 4), we optimize the value of γ using a similar procedure by running gradient descent from the starting value $\frac{\gamma_0}{N^{1/4}}$, where $\gamma_0 \in \{0.1, 0.01, 0.001, 0.0001, 10^{-5}\}$.