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 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.

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 predictions 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 re-designing 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?*

[1]Department of Computer Science, Duke University, Durham NC, United States. Correspondence to: Keerti Anand <kanand@cs.duke.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

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 worst-case ratio between the cost of the online algorithm's solution and that of the (offline) optimum. By leveraging ML predictions, 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 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* (a.k.a. *ski rental*) problem. In this problem, the algorithm is faced with one of two choices: a small recurring (rental) cost, or a large (buying) cost that has to be paid once but no cost thereafter. This choice routinely arises in our daily lives such as in the decision to rent or buy a house, 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 an expected 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 $y|x$ may 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 1) is to design a learning-to-rent algorithm whose competitive ratio is within a factor of $(1+\varepsilon)$ of the best competitive ratio achievable 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 feature dimensions d is exponential, which we show is indeed necessary (Theorem 2).

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 required training samples n to obtain a $(1 + \varepsilon)$ competitive algorithm can be decreased from exponential to linear in d, specifically $O(d/\varepsilon^2)$.

While it's a significant improvement over the previous bound, 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 the dependence on d entirely (although at a slightly worse dependence on ε).

We also consider a noisy model where the labels in the training set are noisy. By this, we mean that labels for a certain fraction of the input distribution are flipped adverserially. We design a noise tolerant algorithm for the learning-to-rent problem with a competitive ratio of $1 + 3\sqrt{p}$, where p is the mis-classification error of a noise tolerant binary classifier. We complement this bound by showing that for a noise level of η, the best competitive ratio achievable is $1 + \frac{\sqrt{\eta}}{2}$.

Next, we consider robustness of our algorithms, i.e., their performance under no assumptions on the input. An important distinction between the recent line of research on online algorithms with predictions and previous "beyond worst case" approaches to competitive analysis is that the recent work simultaneously provides worst case guarantees while also improving the bounds if the additional assumptions on the input hold. Therefore, it is crucial that our algorithms are also robust in this sense. Indeed, we show that in order to obtain a competitive ratio of $(1+\varepsilon)$ in the optimistic scenario, none of our algorithms has competitive ratios any worse than $1+\frac{1}{\varepsilon}$ in the adversarial setting.

Finally, we perform numerical simulations to evaluate our learning-to-rent policies. We consider three different regimes, corresponding to small ($d = 2$), moderate ($d = 100$), and large ($d = 5000$) number of feature dimensions. Recall that our margin-based technique outperforms the black box learning approach for a large number of feature dimensions. This is indeed the case in our experiments: while the two approaches are comparable for $d = 2$ and exhibit relatively mild differences for $d = 100$, the margin-based approach is decidedly superior for $d = 5000$. In principle, this shows that in large instances, there is considerable benefit to customizing ML predictions to make them conducive to the objectives of the online algorithm. In fact, we also show experimentally that although margin-based predictions achieve a smaller competitive ratio, their corresponding mis-classification error is rather large. This provides further evidence that a black box learning approach that simply tries to minimize classification error is not sufficient for generating good predictions for online algorithms. In addition, we also empirically evaluate the performance of our noise-tolerant algorithm and map the competitive ratio as a function of the mis-classification error.

Related Work. A robust literature is beginning to emerge in incorporating ML predictions in online algorithms. While the list of papers in this domain continues to grow by the day, some of the representative problems that this theme has been applied to include: auction pricing (Medina & Vassilvitskii, 2017), rent or buy (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 (originally proposed by 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. (The assumption on the buying cost is w.l.o.g. by appropriate scaling.) Therefore, the optimal offline solution is to buy at the outset when $y \geq 1$ and rent throughout when $y < 1$. We also denote the feature vector by $x \in \mathbb{R}^d$ (e.g., weather predictions, skier behavior, etc.) and assume that (x, y) is drawn from 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 be 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:

$$\text{CR}(\theta, \mathbb{K}) = \mathbb{E}_{(x,y)\sim\mathbb{K}}[g(\theta(x), y)] \tag{1}$$

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

The goal of the learning-to-rent algorithm is to output a function $\theta(\cdot)$ that minimizes 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 $\text{CR}(\theta, \mathbb{K})$ and $\text{CR}(\theta^*, \mathbb{K})$.

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 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.

The additional parameter $\mathbb{K}$ can be dropped when the distribution is clear from the context.

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* (EMD) metric.

Definition 2. *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 EMD as the distance measure between distributions.

Definition 3. *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 $\textsc{emd}(\mathbb{Y}_1, \mathbb{Y}_2) \leq L \cdot \|x_1 - x_2\|_2$.*

Now we are ready to state our main result in this section:

Theorem 1. *For the 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 over 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 $1 \leq i \leq N$.
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 refer the reader to the full version of the paper for a formal analysis of this algorithm.

The main shortcoming of Theorem 1 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:

Theorem 2. *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-

[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.

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 mapping features to labels is restricted to a given *concept class* $\mathcal{C}$:

Definition 4. *Consider a set $X \in \mathbb{R}^d$ and a concept class $\mathcal{C}$ of Boolean functions $X \to \{0,1\}$. Let c be an arbitrary hypothesis 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 $1-\delta$:*

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

Standard results in learning theory show that if the function class $\mathcal{C}$ is "simple", the PAC-learning problem can be solved with a small number of 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 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 i.e $y = f(x)$ for some function f.*

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 a weakening 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 regions $y \geq 1$ 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 the PAC-learning algorithm as a black-box. In other words, 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. Note that we only use Assumption 2 here.

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 relates the competitive ratio achieved by Algorithm 3 with the accuracy of the black-box PAC learner. This implies an upper bound on the sample complexity of learning-to-rent, using the sample complexity bounds for PAC learners.

Theorem 3. *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 ratio of $(1 + 2\sqrt{\epsilon})$ with probability $1-\delta$.*

Remark: The above theorem can be refined for asymmetrical errors (where the classification errors on the two sides are different) showing that the algorithm is more sensitive to errors of one type than the other.

Next, we show that the relationship between PAC-learning and learning-to-rent, established in one direction in Theorem 3, 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 3 is (nearly) optimal without any further assumptions.

Theorem 4. *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.*

4.1. Margin-based PAC-learning for Learning-to-Rent

Theorem 3 is very general in that there are many concept classes for which we have existing PAC-learning bounds. 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. However, the sample complexity can be reduced when the VC-dimension of the concept class is small:

Theorem 5 (e.g., (Kearns & Vazirani, 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 5. *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. This is because $y \approx 1$ for points in the margin, and the competitive ratio of ski rental 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.

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** $\theta(x) = 1 + \gamma$.
```

The filtering process creates an artificial margin:

Lemma 6. *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 7. *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 of $1 + O(L\alpha)$ for n samples with constant failure probability, where α satisfies:*

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

Proof. Let q denote the probability that (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 $\frac{1+\gamma}{1-\gamma}$. The expected competitive ratio is:

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

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 5 and Lemma 6, 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 $1 + O(L\alpha)$. □

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 8 (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:*

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

Feature vectors are typically assumed to be normalized to have constant norm, i.e., $R = O(1)$. Thus, Theorem 7 gives the sampling complexity for linear separators as follows:

Corollary 9. *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.

Corollary 10. *For a kernel function ϕ satisfying $\|\phi(x_1) - \phi(x_2)\| \ge \frac{1}{\nu} \cdot \|x_1 - x_2\|$ 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, the existence of noise-tolerant binary classifiers (e.g., (Blum et al., 1998; Awasthi et al., 2014; Natarajan et al., 2013)), 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.

Algorithm 5 Learning-to-rent with a noisy classifier

```
Set $p_0 = \max(p, \epsilon)$.

Learning:
if $p_0 \le \frac{1}{9(e-1)^2}$
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 $\theta(x) = \sqrt{p_0}$.
```

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:

Theorem 11. *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.

Theorem 12. *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.*

6. Robustness Bounds

In this section, we address the scenario when there is no assumption on the input, i.e., the choice of the input is adversarial. The desirable property in this setting is encapsulated in the following definition of "robustness" adapted from the corresponding notion in (Purohit et al., 2018):

Definition 6. *A learning-to-rent algorithm A with threshold function $\theta(\cdot)$ is said to be γ-robust if $g(\theta(x), y) \le \gamma$ for any feature x and any length of the ski season y.*

First, we show an upper bound on the competitive ratio for any algorithm based on the shortest wait time for any input.

Lemma 13. *A learning-to-rent algorithm 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 approaches θ. This gives us that $\max_{y \in \mathbb{R}^+} g(\theta, y) = \left(1+\frac{1}{\theta}\right)$. Now, since there is no x such that $\theta(x) < \theta_0$, we get:

$$\max_{y \in \mathbb{R}^+, x \in \mathbb{R}^d} g(\theta(x), y) \le \left(1 + \frac{1}{\theta_0}\right). \quad \square$$

The robustness bounds for our algorithms are straightforward applications of the above lemma. We derive these bounds below. First, we consider Algorithm 2 based only on the Lipschitz assumption.

Theorem 14. *Algorithm 2 is $\left(1+\frac{1}{\epsilon}\right)$-robust.*

Proof. Algorithm 2 always chooses a threshold in the range $[\epsilon, 1/\epsilon]$, i.e., $\theta \ge \epsilon$ for all inputs. The theorem now follows by Lemma 13. $\square$

Next, we consider the black box algorithm that uses the PAC learning approach, i.e., Algorithm 3.

Theorem 15. *Algorithm 3 is $\left(1+\frac{1}{\sqrt{\epsilon}}\right)$-robust.*

Proof. Note that Algorithm 3 has $\theta \ge \sqrt{\epsilon}$ for all inputs, which by Lemma 13 gives a robustness bound of $1+\frac{1}{\sqrt{\epsilon}}$. $\square$

Next, we show robustness bounds for the margin-based approach, i.e., Algorithm 4.

Theorem 16. *Algorithm 4 is $\left(1+\frac{1}{L\alpha}\right)$-robust.*

Proof. This follows from Lemma 13, with the observation that the shortest wait time in Algorithm 4 is $\gamma = L\alpha$. $\square$

Finally, we consider the noisy classification setting in Algorithm 5.

Theorem 17. *Algorithm 5 is $\max\left(\frac{e}{e-1}, 1+\frac{1}{\sqrt{\varepsilon}}\right)$-robust.*

Proof. In the two cases in Algorithm 5, either the threshold θ satisfies $\theta \ge \sqrt{p_0}$ or a random threshold is chosen for which the expected competitive ratio is $\frac{e}{e-1}$ for any input. In the first, case, we further note that $p_0 = \max(p, \varepsilon) \ge \varepsilon$, i.e., $1+\frac{1}{\sqrt{p_0}} \le 1+\frac{1}{\sqrt{\varepsilon}}$. The theorem now follows by applying Lemma 13. $\square$

7. 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 $W \in \mathbb{R}^d$ as $W \sim N(0, \mathbf{I}/d)$. 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 $y \ge 1$ when $W^T x \ge 0$ and $y < 1$ otherwise. Note that this satisfies the Lipschitz condition given in Definition 3, with $L = 2$ for $\|W\| \le 1$.

Training and Validation. 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.

The input x is drawn from a mixture distribution, where with probability $1/2$ we sample x from a Gaussian $x \sim N(0, \mathbf{I}/d)$, and with probability $1/2$, we sample x as $x = \alpha W + \eta$, here $\alpha \sim N(0,1)$ is a coefficient in the direction of 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 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.

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 theo-

[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.

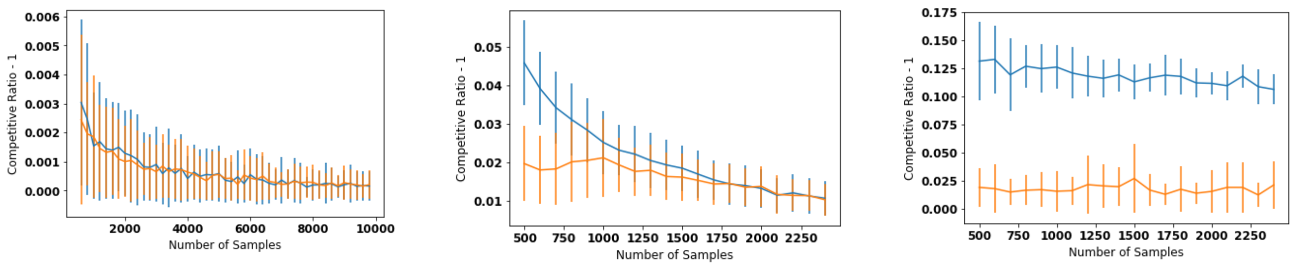

Figure 1. Comparison of Algorithm 3 (blue) and Algorithm 4 (orange). From left to right, $d = 2, 100$, and 5000.

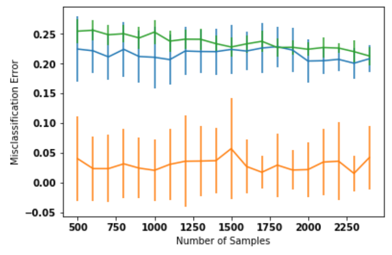

Figure 2. Classification error in Algorithm 3 (green) and Algorithm 4 (blue for all samples, orange for filtered samples).

retical 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 9 and Theorem 3. 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 in the lower bound given by Theorem 12, i.e, $\mathbb{P}[y = z] = \frac{e}{e-1} \cdot z \cdot e^{-z}$ 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)

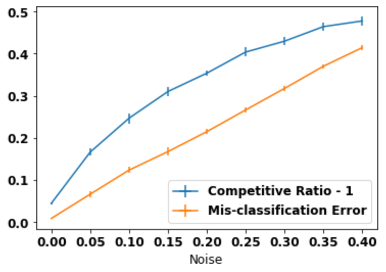

Figure 3. Algorithm 5 with varying noise rate with $d = 100$.

8. Conclusion and Future Work

In this paper, we explored the question of customizing machine 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 (which is responsible for large sample complexity if accurate classification were the end goal). In addition, we showed worst-case robustness bounds for our algorithms, i.e., that they exhibit bounded competitive ratios even if the input is adversarial.

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.

Acknowledgments

The authors thank the anonymous reviewers for their helpful comments and feedback. This work was done under the auspices of the Indo-US Virtual Networked Joint Center on Algorithms under Uncertainty. K. Anand and D. Panigrahi were supported in part by NSF grants CCF-1535972, CCF-1955703, and an NSF CAREER Award CCF-1750140. R. Ge was 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.

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. and Vazirani, U. V. *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.