Learning Predictions for Algorithms with Predictions

Anonymous Author(s)
Affiliation
Address
email

Abstract

A burgeoning paradigm in algorithm design is the field of *algorithms with predictions*, in which algorithms are designed to take advantage of a possibly-imperfect prediction of some aspect of the problem. While much work has focused on using predictions to improve competitive ratios, running times, or other performance measures, less effort has been devoted to the question of how to obtain the predictions themselves, especially in the critical online setting. We introduce a general design approach for algorithms that learn predictors: (1) identify a functional dependence of the performance measure on the prediction quality, and (2) apply techniques from online learning to learn predictors against adversarial instances, tune robustness-consistency trade-offs, and obtain new statistical guarantees. We demonstrate the effectiveness of our approach at deriving learning algorithms by analyzing methods for bipartite matching, ski-rental, page migration, and job scheduling. In the first two settings we improve upon existing learning-theoretic results by deriving online results, obtaining better or more general statistical guarantees, and utilizing a much simpler analysis, while in the last two we provide the first learning-theoretic guarantees.

1 Introduction

Algorithms with predictions, a subfield of beyond-worst-case analysis of algorithms [30], aims to design methods that make use of machine-learned predictions in order to reduce runtime, error, or some other performance cost. Mathematically, given some prediction $\mathbf{x}$, algorithms in this field are designed such that the cost $C_t(\mathbf{x})$ of an instance t while using the prediction is upper-bounded by some measure $U_t(\mathbf{x})$ of the quality of the prediction on that instance. The canonical example here is that the cost of binary search on a sorted array of size n can be improved from $\mathcal{O}(\log n)$ to at most $U_t(\mathbf{x}) = 2\log \eta_t(\mathbf{x})$, where $\eta_t(\mathbf{x})$ is the distance between the true location of a query t in the array and the location predicted by the predictor $\mathbf{x}$ [30]. In recent years, algorithms whose cost depends on the quality of possibly imperfect predictions have been developed for numerous important problems, including caching [33, 19, 28], scheduling [25, 35], ski-rental [24, 1, 12], bipartite matching [13], page migration [17], and many more [9, 14, 30].

While there has been a significant effort to develop algorithms that can take advantage of learned predictions, there has been less focus on actually *learning* to predict. For example, of the works listed above only two focusing on ski-rental [1, 12] and one other [13] show sample complexity guarantees, and none consider the important *online learning* setting, in which problem instances are not guaranteed to come from a fixed distribution. This is in contrast to the related area of data-driven algorithm design [16, 3], which has established techniques such as dispersion for deriving learning-theoretic guarantees, leading to end-to-end results encompassing both learning and computation [6]. It is also despite the fact that, as we see in this work, learning even simple predictors is in many cases a non-trivial problem.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

Table 1: Problems we apply our framework to, new learning algorithms we derive, and their regret.

Problem	Algorithm with prediction	Feedback	Upper bound	Online algo.	Regret
Min. wt. bipartite matching (3)†	Hungarian method initialized by dual $\hat{\mathbf{x}} \in \mathbb{R}^n$	Opt. dual $\mathbf{x}^*(\mathbf{c})$	$\mathcal{O}(\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1)$	Online gradient	$\mathcal{O}\left(n\sqrt{nT}\right)$
Min. weight $\mathbf{b}$-matching (B)†	Hungarian method initialized by dual $\hat{\mathbf{x}} \in \mathbb{R}^n$	Opt. dual $\mathbf{x}^*(\mathbf{c}, \mathbf{b})$	$\mathcal{O}(\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1})$	Online gradient	$\mathcal{O}\left(n\sqrt{nT}\right)$
Online page migration (4)†	Lazy offline optimal for predictions $\{\hat{s}_{[j]} \sim \mathbf{p}_{[j]}\}_{j=1}^n$	Requests $\{s_{[j]}\}_{j=1}^n$	$\tilde{\mathcal{O}}\left(\max_{i\in[n]} \mathbb{E}_{\mathbf{p}} \sum_{j=i}^{i+\gamma D} 1_{\hat{s}_{[j]} \neq s_{[j]}}\right)$	Exponent. gradient $\times\, n$	$\mathcal{O}\left(\sqrt{nT}\right)$
Non-clairvoyant job scheduling (5)	Preferential round-robin with trade-off parameter λ	Prediction quality η	$\min\left\{\frac{1+2\eta/n}{1-\lambda}, \frac{2}{\lambda}\right\}$	Continuum Hedge algo.	$\mathcal{O}\left(\sqrt{T\log T}\right)$
Ski-rental w. integer days $n \in [N]$ (5)	Buy if price $b \leq x$, λ trade-off with worst-case approx.	Number of ski-days n	$\frac{\min\{\lambda(b1_{x>b}+n1_{x\leq b}),b,n\}}{1-(1+1/b)^{-b\lambda}}$	Hedge algo.	$\mathcal{O}(N\sqrt{T\log(NT)})$
Ski-rental with β-dispersed n (5)	Buy after x days, λ trade-off with worst-case approx.	Number of ski-days n	$\min\left\{\frac{e\min\{n,b\}}{(e-1)\lambda}, \frac{n1_{n\leq x}+(b+x)1_{n>x}}{1-\lambda}\right\}$	Exponential forecaster	$\mathcal{O}\left(\sqrt{T\log(NT)} + N^2T^{1-\beta}\right)$

† For these algorithms we also provide new statistical guarantees in the i.i.d. setting.

We bridge this gap and provide a framework for obtaining learning-theoretic guarantees for algorithms with predictions. In addition to improving sample complexity bounds, we show how to learn the parameters of interest in an *online* setting, and prove bounds on the overall regret. We accomplish this using a two-step approach inspired by recent work on theoretical meta-learning [20], which has been used to derive numerous multi-task learning results by optimizing regret-upper-bounds that encode the task-similarity [26, 27, 8, 21]. As evidenced by our results in Table 1, we believe the two-step framework below holds similar potential for obtaining guarantees for algorithms with predictions.

Our framework is as follows:

1. For a given algorithm, derive a convenient-to-optimize upper bound $U_t(\mathbf{x})$ on the cost $C_t(\mathbf{x})$ that depends on both the prediction $\mathbf{x}$ and information specific to instance t returned once the algorithm terminates, e.g. the optimum in combinatorial optimization. We find that in many cases such bounds already exist, and the quality of the prediction can be measured by a distance from some ground truth obtained from the output, a quantity that is usually convex and thus learnable.
2. Apply online learning to obtain both regret guarantees against adversarial sequences and sample complexity bounds for i.i.d. instances. We provide pseudo-code for a generic setup in Algorithm 1.

Table 1 summarizes instantiations of our framework on multiple problems. Our approach is designed to be simple-to-execute, leaving much of the difficulty to what the field is already good at: designing algorithms and proving prediction-quality-dependent upper bounds on their costs. Once the latter is accomplished, our framework leverages problem-specific structure to design a customized learning algorithm for each problem, leading to strong regret and sample complexity guarantees. In particular, in two of the four specific problems we study we improve upon existing guarantees in either sample complexity or generality, while in the other two we show the first learning-theoretic guarantees whatsoever. In all cases we are the first to show regret guarantees in the online setting. This demonstrates the usefulness of and need for such a theoretical framework for studying these problems.

We summarize the diverse set of contributions enabled by our theoretical framework below:

1. **Bipartite matching:** Our starting example builds upon the work on **minimum-weight bipartite matching** using the Hungarian algorithm by Dinitz et al. [13]. We show how our framework leads directly to algorithms that yield both the first regret guarantees in the online setting and new sample complexity bounds that improve slightly over the previous approach. In the Appendix we show similar improvements for $\mathbf{b}$-matching; in-fact, our guarantees cover the more general case of demand vectors $\mathbf{b}$ chosen i.i.d. or adversarially, unlike those of Dinitz et al. [13] that keep it fixed.
2. **Page migration:** We next study a more challenging application, **online page migration**, and show how we can adapt the algorithmic guarantee of Indyk et al. [17] into a learnable upper bound for which we can again provide both adversarial and statistical guarantees.
3. **Tuning robustness-consistency trade-offs:** Many bounds for *online* algorithms with predictions incorporate parameterized trade-offs between trusting the prediction completely or falling back on a worst-case approximation. The structure of these trade-offs lends itself to online tuning of the trade-off parameter, which we instantiate on a simple **job scheduling** problem with a fixed predictors. Then we turn to the more challenging problem of simultaneously tuning these parameters and learning predictions, which we achieve on two variants of the **ski-rental** problem. For the discrete case we give the only learning-theoretic guarantee, while for the continuous case our regret bound makes a dispersion assumption [6] that, in the i.i.d. setting, is a strictly weaker assumption than the log-concave requirement of Diakonikolas et al. [12].

2 Related work

Algorithms with predictions is a type of beyond-worst-case analysis of algorithms [34]; along with areas like smooth analysis [37] and data-driven algorithm design [3], it takes advantage of the fact that real-world instances are not worst-case. Inspired by success in applications such as learned indices [22], there has been a great deal of theoretical study focusing on algorithms whose guarantees depend on the quality of a given predictor (c.f. the Mitzenmacher and Vassilvitskii [30] survey). The actual learning of this predictor has been studied less [1, 12, 13] and rarely in the online setting; we aim to change this with our study. Some papers improve online learning itself using predictions [31, 18, 11], nut they also assume known predictors or only learn over a small set of policies, and their goal is minimizing regret not computation. In-general, we focus on showing how algorithms with predictions can make use of online learning rather than on new methods for the latter. Several works [4, 32, 2] use learning *while* advising an algorithm, in-effect taking a learning-inspired approach to better make use of a prediction *within* an algorithm, whereas we focus on learning the prediction *outside* of the target algorithm. Our paper presents the first general framework for efficiently learning useful predictors.

Data-driven algorithm design is a related area that has seen more learning-theoretic effort [16, 6, 3]. At a high-level, it often studies tuning parameters such as the gradient descent step-size [16] or settings of branch and bound [5], whereas the predictors in algorithms with predictions guess the sequence in an online algorithm [17] or the actual outcome of the computation [13]. The distinction can be viewed as terminological, since a prediction can be viewed as a parameter, but it can mean that in our settings we have full information about the loss function since it is typically some discrepancy between the full sequence or computational outcome and the prediction. In contrast, in data-driven algorithm design getting the cost of each parameter often requires additional computation, leading to (semi-)bandit settings [7]. A more salient difference is that data-driven algorithm design guarantees compete with the parameter that minimizes average cost but do not always quantify the improvement attainable via learning; in algorithms with predictions we do generally quantify this improvement with an upper bound on the cost that depends on the prediction quality, but we usually only compete with the parameter that is optimal for prediction quality, which is not always cost-optimal. We do adapt data-driven algorithm design tools like dispersion [6] for algorithms with predictions.

Our two-step approach to providing guarantees for algorithms with predictions is inspired by the Average Regret-Upper-Bound Analysis (ARUBA) framework [20] for studying meta-learning [15]. Instead of instances they have tasks with different data-points, and the upper bounds are on learning-theoretic quantities such as regret rather than computational costs such as runtime. Mathematically, ARUBA takes advantage of similar structure in the regret-upper-bounds that we find in algorithms with predictions, namely that the upper bounds encode some measure of the quality of the prediction (in their case an initialization for gradient descent) via a comparison to the ground-truth (in their case the optimal parameter). However, whereas in ARUBA the need to know this optimal parameter after seeing a task is a weakness that does not hold in practice, in algorithms with predictions the corresponding quantity—the feedback listed in Table 1—is generally known after seeing the instance.

3 Framework overview and application to bipartite matching

In this section we outline the theoretical framework for designing algorithms and proving guarantees for learned predictors. As an illustrative example we will use the Hungarian algorithm for bipartite matching, for which Dinitz et al. [13] demonstrated an instance-dependent upper bound on the running time using a learned dual vector. Along the way, we will show an improvement to their sample complexity bound together with the first online results for this setting.

Bipartite matching: For a bipartite graph on n nodes and m edges, **min-weight perfect matching** (MWPM) asks for the perfect matching with the least weight according to edge-costs $\mathbf{c} \in \mathbb{Z}^m_{\geq 0}$. A common approach here is the Hungarian algorithm, a convex optimization-based approach for which Dinitz et al. [13] showed a runtime bound of $\tilde{\mathcal{O}}\left(m\sqrt{n}\min\left\{\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1, \sqrt{n}\right\}\right)$, where $\hat{\mathbf{x}} \in \mathbb{Z}^n$ initializes the duals in a primal-dual algorithm and $\mathbf{x}^*(\mathbf{c}) \in \mathbb{Z}^n$ is dual of the optimal solution; note that the latter is obtained for free after running the Hungarian method.

Step 1 -Upper bound: The first step of our approach is to find a suitable function $U_t(\mathbf{x})$ of the prediction $\mathbf{x}$ that (a) upper bounds the target algorithm's cost $C_t(\mathbf{x})$, (b) can be constructed completely once the algorithm terminates, and (c) can be efficiently optimized. These qualities allow learning the predictor in the second step. The requirements are similar to those of ARUBA for showing results for meta-learning [20], although there the quantity being upper-bounded was regret, not algorithmic cost.

Algorithm 1: Generic pseudo-code for using an online learning algorithm over $\mathcal{X}$ to learn a predictor for a method AlgorithmWithPrediction that takes advice from $\mathcal{X}$ and returns upper bounds U_t on its cost. The goal of OnlineAlgorithm is low regret over the sequence U_t, so that on-average C_t is upper bounded by the smallest possible average of $U_t(\mathbf{x})$, up to some error decreasing in T. For specific instantiations of both algorithms and the way they provide feedback see Table 1.

initialize $\hat{\mathbf{x}}_1 \in \mathcal{X}$ for OnlineAlgorithm
for *instance* $t = 1, \ldots, T$ **do**
  obtain instance I_t
  run AlgorithmWithPrediction$(I_t, \hat{\mathbf{x}}_t)$
    suffer cost $C_t(\hat{\mathbf{x}}) \leq U_t(\hat{\mathbf{x}}_t)$
    get feedback to construct upper bound U_t
  $\hat{\mathbf{x}}_{t+1} \leftarrow$ OnlineAlgorithm$(U_t, \hat{\mathbf{x}}_t)$

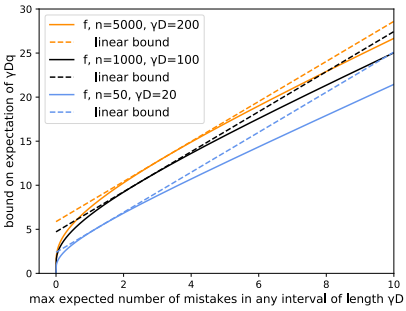

Figure 1: Plot of the upper bound f (c.f. Lemma 4.1) on the expected maximum number of mistakes in any interval of length γD as a function of the maximum expected number, together with its own linear upper bound that we use, for three settings of $n, \gamma D$.

Many guarantees for algorithms with predictions are already amenable to being optimized, although we will see that they can require some massaging in order to be useful. In many cases the guarantee is a distance metric between the prediction $\hat{\mathbf{x}}$ and some instance-dependent perfect prediction $\mathbf{x}^*$, which is convex and thus straightforward to learn. This is roughly true of our bipartite matching example, although taking the minimum of a constant and the distance $\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1$ between the predicted and actual duals makes the problem non-convex. However, we can further upper bound their result by $\tilde{\mathcal{O}}\left(m\sqrt{n}\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1\right)$; note that Dinitz et al. [13] also optimize this quantity, not the tighter upper bound with the minimum. While this might seem to be enough for step one, Dinitz et al. [13] also require the prediction $\hat{\mathbf{x}}$ to be integral, which is difficult to combine with standard online procedures. In order to get around this issue, we show that rounding any nonnegative real vector to the closest integer vector incurs only a constant multiplicative loss in terms of the ℓ_1-distance.

Claim 3.1. *Given any vectors $\mathbf{x} \in \mathbb{Z}^n$ and $\mathbf{y} \in \mathbb{R}^n$, let $\tilde{\mathbf{y}} \in \mathbb{Z}^n$ be the vector whose elements are those of $\mathbf{y}$ rounded to the nearest integer. Then $\|\mathbf{x} - \tilde{\mathbf{y}}\|_1 \leq 2\|\mathbf{x} - \mathbf{y}\|_1$.*

Proof. Let $S \subset [n]$ be the set of indices $i \in [n]$ for which $\mathbf{x}_{[i]} \geq \mathbf{y}_{[i]} \iff \tilde{\mathbf{y}}_{[i]} = \lceil \mathbf{y}_{[i]} \rceil$. For $i \in [n] \backslash S$ we have $|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| \geq 1/2 \geq |\tilde{\mathbf{y}}_{[i]} - \mathbf{y}_{[i]}|$ so it follows by the triangle inequality that

$$\begin{aligned}\|\mathbf{x} - \tilde{\mathbf{y}}\|_1 = \sum_{i \in S} |\mathbf{x}_{[i]} - \tilde{\mathbf{y}}_{[i]}| + \sum_{i \in [n] \backslash S} |\mathbf{x}_{[i]} - \tilde{\mathbf{y}}_{[i]}| &\leq \sum_{i \in S} |\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + \sum_{i \in [n] \backslash S} |\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + |\mathbf{y}_{[i]} - \tilde{\mathbf{y}}_{[i]}| \\ &\leq \sum_{i \in S} |\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + 2 \sum_{i \in [n] \backslash S} |\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| \leq 2\|\mathbf{x} - \mathbf{y}\|_1\end{aligned}$$

□

Combining this projection with the convex relaxation above and the result of Dinitz et al. [13] shows that for any predictor $\mathbf{x} \in \mathbb{R}^n$ we have (up to affine transformation) a convex upper bound $U_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_t^*\|_1$ on the runtime of the Hungarian method, as desired. We now move to step two.

Step 2 - Online learning: The second component of our approach is natural: once one has an upper bound on the cost, apply standard online learning algorithms and results to these upper bounds to obtain guarantees for learning predictions. For example, for the bipartite matching we can just apply regular projected online gradient descent (OGD) to the sequence of losses $U_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t)\|_1$; as shown in Theorem 3.1, this yields a regret guarantee via a textbook result. The simplicity here is the point; by relegating as much of the difficulty as we can to obtaining an easy-to-optimize upper bound in step one, we make the actual learning-theoretic component easy. However, as we show in the following sections, it is not always easy to obtain a suitable upper bound, nor is it always obvious what online learning algorithm to apply, e.g. if the upper bounds are non-convex.

Our use of online learning for this step is motivated by two factors:

1. Being able to do well on non-i.i.d. instances is important in practical applications, e.g. in job scheduling problems where demand for computational resources is constantly changing over time.
2. There are classic procedures for converting regret guarantees into sample complexity bounds via online-to-batch conversion [10]. As shown in Theorem 3.1, for the bipartite matching example this conversion leads to a slightly *better* sample complexity than that of Dinitz et al. [13].

We now show how this second step is implemented for the bipartite matching problem by improving upon the result of Dinitz et al. [13] in Theorem 3.1 below; the specific improvement is in the entirely new regret guarantee against adversarial cost vectors and a $\mathcal{O}(\log n)$ improvement in the sample complexity. Note how the proof requires only their existing algorithmic contribution, Claim 3.1, and some standard tools in online convex optimization.

Theorem 3.1. *Suppose we have a fixed bipartite graph with n vertices and m edges.*

1. *For any cost vector $\mathbf{c} \in \mathbb{Z}^m_{\geq 0}$ and any dual vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ there exists an algorithm for MWPM that runs in time*
$$\tilde{\mathcal{O}}\left(m\sqrt{n}\min\left\{U(\hat{\mathbf{x}}), \sqrt{n}\right\}\right) \leq \tilde{\mathcal{O}}\left(m\sqrt{n}U(\hat{\mathbf{x}})\right)$$
for $U(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*(\mathbf{c})\|_1$, where $\mathbf{x}^*(\mathbf{c})$ the optimal dual vector associated with $\mathbf{c}$.
2. *For any $\delta, \varepsilon > 0$ and any distribution $\mathcal{D}$ over integer m-vectors with ℓ_∞-norm $\leq C$ there exists a poly-time algorithm that takes $\mathcal{O}\left(\left(\frac{nC}{\varepsilon}\right)^2\left(n + \log\frac{1}{\delta}\right)\right)$ samples from $\mathcal{D}$ and returns $\hat{\mathbf{x}}$ s.t. w.p. $\geq 1 - \delta$:*
$$\mathbb{E}_{\mathbf{c}\sim\mathcal{D}}\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1 \leq \min_{\|\mathbf{x}\|_\infty \leq C} \mathbb{E}_{\mathbf{c}\sim\mathcal{D}}\|\mathbf{x} - \mathbf{x}^*(\mathbf{c})\|_1 + \varepsilon$$
3. *Let $\mathbf{c}_1, \ldots, \mathbf{c}_T \in \mathbb{Z}^m_{\geq 0}$ be an adversarial sequence of m-vectors with ℓ_∞-norm $\leq C$. There exists an efficient algorithm that predicts $\hat{\mathbf{x}}_t$ after seeing $\mathbf{c}_1, \ldots, \mathbf{c}_{t-1}$ and achieves regret*
$$\max_{\|\mathbf{x}\|_\infty \leq C} \sum_{t=1}^{T} \|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c}_t)\|_1 - \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t)\|_1 \leq Cn\sqrt{2nT}$$

Proof. The first result follows by combining Dinitz et al. [13, Theorem 13] with Claim 3.1. For the third result, let $\hat{\mathbf{x}}_t$ be the sequence generated by running OGD [38] with step size $\frac{C}{\sqrt{2nT}}$ on the losses $U_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t)\|_1$ over domain $[-C, C]^n$. Since these losses are n-Lipschitz and the duals are $C\sqrt{n}$-bounded in Euclidean norm the regret guarantee follows from Shalev-Shwartz [36, Corollary 2.7]. For the second result, we apply standard online-to-batch conversion to the third result, i.e. we draw $T = \Omega\left(\left(\frac{nC}{\varepsilon}\right)^2\left(n + \log\frac{1}{\delta}\right)\right)$ samples $\mathbf{c}_t$, run OGD on the resulting losses $U_t(\mathbf{x})$, and set $\hat{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^{T}\hat{\mathbf{x}}_t$ to be the average of the resulting predictions $\hat{\mathbf{x}}_t$. Then applying Jensen's inequality, Cesa-Bianchi et al. [10, Proposition 1], the regret guarantee above, and Hoeffding's inequality yields

$$\begin{aligned}
\mathbb{E}_{\mathbf{c}}\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c})\|_1 \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\mathbf{c}}\|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c})\|_1 &\leq \frac{1}{T}\sum_{t=1}^{T}\|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c}_t)\|_1 + Cn\sqrt{\frac{2}{T}\log\frac{2}{\delta}} \\
&\leq \min_{\|\mathbf{x}\|_\infty \leq C}\frac{1}{T}\sum_{t=1}^{T}\|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t)\|_1 + \frac{Cn\sqrt{2n} + Cn\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}} \\
&\leq \min_{\|\mathbf{x}\|_\infty \leq C}\mathbb{E}_{\mathbf{c}}\|\mathbf{x} - \mathbf{x}^*(\mathbf{c})\|_1 + \frac{Cn\sqrt{2n} + 2Cn\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}}
\end{aligned}$$

□

This concludes an overview of our two-step approach for obtaining learning guarantees for algorithms with predictions. To summarize, we propose to (1) obtain simple-to-optimize upper bounds $U_t(\mathbf{x})$ on the cost of the target algorithm on instance t as a function of prediction $\mathbf{x}$ and (2) optimize $U_t(\mathbf{x})$ using online learning. While conceptually simple, even in this illustrative example it already improves upon past work; in the sequel we demonstrate further results that this approach makes possible. Not that, like Dinitz et al. [13], we are also able to generalize Theorem 3.1 to $\mathbf{b}$-matchings, which we do in Appendix B; another advantage of our approach is that it lets us prove online and statistical learning in the case where the demand vector $\mathbf{b}$ varies across instances rather than staying fixed as in Dinitz et al. [13].

4 Predicting requests for page migration

Equipped with our two-step approach for deriving guarantees for learning predictors, we investigate several more important problems in combinatorial optimization, starting with the page migration problem. Our results demonstrate that even for learning such simple predictors there are interesting technical challenges in deriving a *learnable* upper bound. Nevertheless, once this is accomplished the second step of our approach is again straightforward.

Page migration: Consider a server that sees a sequence of requests $s_{[1]}, \dots, s_{[n]}$ from metric space $(\mathcal{K}, d)$ and at each timestep decides whether to change its state $a_{[i]} \in \mathcal{K}$ at cost $Dd(a_{[i-1]}, a_{[i]})$ for some $D > 1$; it then suffers a further cost $d(a_{[i]}, s_{[i]})$. The **online page migration** (OPM) problem is then to minimize the cost to the server. Recently, Indyk et al. [17] studied a setting where we are given a sequence of predicted points $\hat{s}_{[1]}, \dots, \hat{s}_{[n]} \in \mathcal{K}$ to aid the page migration algorithm. They show that if there exists $\gamma, q \in (0, 1)$ s.t. $\gamma D \in [n]$ and for any $i \in [n]$ we have $\sum_{j=i}^{i+\gamma D-1} 1_{s_{[j]} \neq \hat{s}_{[j]}} \leq q\gamma D$ then there exists an algorithm with competitive ratio $(1 + \gamma)(1 + \mathcal{O}(q))$ w.r.t. to the offline optimal. This algorithm depends on γ but not q, so we study the setting where γ is fixed.

Deriving an upper bound: As in the previous section, the predictions are discrete, so to use our approach we must convert it into a continuous problem. As we have fixed γ, the competitive ratio is an affine function of the following upper bound on q:

$$Q(\hat{s}, s) = \frac{1}{\gamma D} \max_{i \in [n-\gamma D+1]} \sum_{j=i}^{i+\gamma D-1} 1_{\hat{s}_{[j]} \neq s_{[j]}}$$

We assume that the set of points $\mathcal{K}$ is finite with indexing $k = 1, \dots, |\mathcal{K}|$ and use this to introduce our continuous relaxation, a natural randomized approach converting the problem of learning a prediction into n experts problems on $|\mathcal{K}|$ experts. For each $j \in [n]$ we define a probability vector $\mathbf{p}_{[j]} \in \triangle_{|\mathcal{K}|}$ governing the categorical r.v. $\hat{s}_{[j]}$, i.e. $\Pr\{\hat{s}_{[j]} = k\} = \mathbf{p}_{[j,k]}\ \forall\ k \in \mathcal{K}$. Under these distributions the expected competitive ratio will be $(1 + \gamma)(1 + \mathcal{O}(\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q(\hat{s}, s)))$, for $\mathbf{p}$ the product distribution of the vectors $\mathbf{p}_j$. Note that forcing each $\mathbf{p}_j$ to be a one-hot vector recovers the original approach with no loss, so optimizing $\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q(\hat{s}, s)$ over $\mathbf{p} \in \triangle_{|\mathcal{K}|}^n$ would find a predictor that fits the original result.

However, $\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q(\hat{s}, s)$ is not convex in $\mathbf{p}$. The simplest relaxation is to replace the maximum by summation, but this leads to a worst-case bound of $\mathcal{O}\left(\frac{n}{\gamma D}\right)$. We instead bound $\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q(\hat{s}, s)$—and thus also the expected competitive ratio—by a function of the following maximum over expectations:

$$U_s(\mathbf{p}) = \max_{i \in [n-\gamma D+1]} \mathbb{E}_{\hat{s} \sim \mathbf{p}} \sum_{j=i}^{i+\gamma D-1} 1_{\hat{s}_{[j]} \neq s_{[j]}} = \max_{i \in [n-\gamma D+1]} \sum_{j=i}^{i+\gamma D-1} 1 - \langle \mathbf{s}_{[j]}, \mathbf{p}_{[j]} \rangle$$

where $\mathbf{s}_{[j,k]} = 1_{s_{[j]}=k}\ \forall\ k \in \mathcal{K}$, i.e. $\mathbf{s}_{[j]}$ encodes the location in $\mathcal{K}$ of the jth request. As a maximum over $n - \gamma D + 1$ convex functions this objective is also convex. Note also that if $U_s(\mathbf{p})$ is zero—i.e. the probability vectors are one-hot and perfect—then $\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q(\hat{s}, s) \geq q$ will also be zero. In fact, q is upper-bounded by a monotonically increasing function of $U_s(\mathbf{p})$, but as this function is concave and non-Lipschitz (c.f. Figure 1) we incur an additive $\mathcal{O}\left(\frac{\log(n-\gamma D+1)}{\gamma D}\right)$ loss to obtain an online-learnable upper bound. This is formalized in the following result (proof in A.1).

Lemma 4.1. *There exist constants $a < e, b < 2/e$ and a monotonically increasing function $f : [0, \infty) \mapsto [0, \infty)$ s.t. $f(0) = 0$ and*

$$\mathbb{E}_{\hat{s} \sim \mathbf{p}} Q_\gamma(\hat{s}, s) \leq \frac{f(U_s(\mathbf{p}))}{\gamma D} \leq \frac{aU_s(\mathbf{p}) + b \log(n - \gamma D + 1)}{\gamma D}$$

We now have an convex bound on the competitive ratio for the OPM algorithm of Indyk et al. [17]. For both this and bipartite matching we resorted to a relaxation of a discrete problem. However, whereas before we only incurred a multiplicative loss (c.f. Claim 3.1), here we have an additive loss that makes the bound meaningful only for $\gamma D \gg \log n$. However, as the method we propose optimizes $U_s(\mathbf{p})$, which bounds q with no additive error via the function f in Lemma 4.1, in-practice we may expect it to help minimize q in all regimes. Note that the non-Lipschitzness near zero that prevents using f for formal regret guarantees comes from the poor tail behavior of Poisson-like random variables with small means, which we do not expect can be significantly improved.

Learning guarantees: Having established an upper bound, in Theorem 4.2 we again show how a learning-theoretic result follows from standard online learning. This time, instead of OGD we run the exponentiated gradient (EG) algorithm [36], a classic approach to learning from experts, on each of the n simplices to learn the probability vectors $\mathbf{p}_{[j]}$. EG performs updates by multiplying by the negative gradient and is notable for having regret logarithmic in the size $|\mathcal{K}|$ of the simplices, which is important if the metric space is large. Note that as the relaxation is randomized, our algorithms output a dense probability vector; to obtain a prediction for OPM we must sample $\hat{s}_{t_{[j]}} \sim \mathbf{p}_{[j]} \; \forall \; j \in [n]$.

Theorem 4.2. *Let $(\mathcal{K}, d)$ be a finite metric space.*

1. *For any request sequence s and any set of probability vectors $\mathbf{p} \in \triangle^n_{|\mathcal{K}|}$ there exists an algorithm for OPM with expected competitive ratio*

$$(1+\gamma)\left(1+\mathcal{O}\left(\frac{U_s(\mathbf{p})+\log(n-\gamma D+1)}{\gamma D}\right)\right)$$

2. *For any $\delta, \varepsilon > 0$ and distribution $\mathcal{D}$ over request sequence of length n in $\mathcal{K}$ there exists a poly-time algorithm that takes $\mathcal{O}\left(\left(\frac{\gamma D}{\varepsilon}\right)^2 n \log \frac{|\mathcal{K}|}{\delta}\right)$ samples from $\mathcal{D}$ and returns $\hat{\mathbf{p}}$ s.t. w.p. $\geq 1-\delta$:*

$$\mathbb{E}_{s\sim\mathcal{D}} U_s(\hat{\mathbf{p}}) \leq \min_{\mathbf{p}\in\triangle^n_{|\mathcal{K}|}} \mathbb{E}_{s\sim\mathcal{D}} U_s(\mathbf{p}) + \varepsilon$$

3. *Let $s_1, \ldots, s_T$ be an adversarial sequence of request sequences. There exists an efficient algorithms that plays $\mathbf{p}_t$ after seeing $s_1, \ldots, s_{t-1}$ and has regret*

$$\max_{\mathbf{p}\in\triangle^n_{|\mathcal{K}|}} \sum_{t=1}^{T} U_{s_t}(\mathbf{p}_t) - U_{s_t}(\mathbf{p}) \leq \gamma D\sqrt{2nT\log|\mathcal{K}|}$$

Proof. The first result follows by combining Indyk et al. [17, Theorem 1] with Lemma 4.1. For the third result let $\mathbf{p}_t$ be the sequence generated by running n randomized exponentiated gradient (EG) algorithms with step size $\sqrt{\frac{\log|\mathcal{K}|}{2\gamma^2 D^2 T}}$ on the losses $U_{s_t}(\mathbf{p})$ over $\triangle^n_{|\mathcal{K}|}$. Since these losses are γD-Lipschitz and the maximum entropy over the simplex is $\log|\mathcal{K}|$, the regret guarantee follows from [36, Theorem 2.15]. For the second result, we apply standard online-to-batch conversion to the third result, i.e. we draw $T = \Omega\left(\left(\frac{\gamma D}{\varepsilon}\right)^2 \log\frac{|\mathcal{K}|}{\delta}\right)$ samples $\mathbf{s}_t$, run EG on the resulting losses $U_{s_t}(\mathbf{p})$ as above, and set $\hat{\mathbf{p}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{p}_t$ to be the average of the resulting actions $\mathbf{p}_t$. Then applying Jensen's inequality, Cesa-Bianchi et al. [10, Proposition 1], the regret guarantee above, and Hoeffding's inequality yields

$$\begin{aligned}
\mathbb{E}_s U_s(\hat{\mathbf{p}}) \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_s U_s(\hat{\mathbf{p}}) &\leq \frac{1}{T}\sum_{t=1}^{T} U_{s_t}(\mathbf{p}_t) + \gamma D\sqrt{\frac{2}{T}\log\frac{2}{\delta}} \\
&\leq \min_{\mathbf{p}\in\triangle^n_{|\mathcal{K}|}} \frac{1}{T}\sum_{t=1}^{T} U_{s_t}(\mathbf{p}) + \frac{\gamma D\sqrt{2n\log|\mathcal{K}|} + \gamma D\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}} \\
&\leq \min_{\mathbf{p}\in\triangle^n_{|\mathcal{K}|}} \mathbb{E}_s U_s(\mathbf{p}) + \frac{\gamma D\sqrt{2n\log|\mathcal{K}|} + 2\gamma D\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}}
\end{aligned}$$

□

As before, this result first shows how the quantity of interest—in this case the competitive ratio—is upper-bounded by an affine function of some measure $U_s(\mathbf{p})$, for which we then provide regret and statistical guarantees using online learning. The difficulty deriving a suitable upper bound exemplifies the technical challenges that arise in learning predictors, difficulties that may be encountered again in other sequence prediction problems such as TCP [9]. Nevertheless, our approach does yield an online procedure that incurs only $\mathcal{O}(\frac{\log n}{\gamma D})$ additive error over Indyk et al. [17] in the case of a perfect predictor and, unlike their work, we provide an algorithm for learning the predictor itself.

5 Tuning robustness-consistency trade-offs and ski-rental

We turn to tuning robustness-consistency trade-offs, introduced in Lykouris and Vassilvitskii [28]. This trade-off captures the tension between following the predictions when they are good (consistency) and doing not much worse than the worst-case guarantee in either case (robustness). In many cases, this trade-off can be made explicit, by a parameter $\lambda \in [0, 1]$. The setting of λ is crucial, yet previous work left the decision on the value of λ to the end-user. Here we show that it is often eminently learnable in an online setting, achieving low regret guarantees.

Then we demonstrate how to accomplish a much harder task—tuning λ at the same time as learning to predict—on two related but technically very different variants of the ski-rental problem. This meta-application highlights the applicability of our approach to non-convex upper bounds.

Robustness-consistency trade-offs: Most problems studied in online algorithms with predictions usually have existing worst-case guarantees on the competitive ratio, i.e. a constant $\gamma \geq 1$ on how much worse (multiplicatively) a learning-free algorithm does relative to the offline optimal cost OPT_t on instance t. While the goal of algorithms with predictions is to use data to do better than this worst-case bound, an imperfect prediction may lead to performance that is much worse. As a result, most guarantees strive to upper bound the cost of an algorithm with prediction $\mathbf{x}$ on an instance t in the following manner:

$$C_t(\mathbf{x}, \lambda) \leq \min\{f(\lambda)u_t(\mathbf{x}), g_t(\lambda)\}$$

Here u_t is some measure of the quality of $\mathbf{x}$ on instance t, $\lambda \in [0, 1]$ is a parameter, f is a monotonically increasing function that ideally satisfies $f(0) = 1$, and g_t is a monotonically decreasing function that ideally satisfies $g_t(1) = \gamma\text{OPT}_t$. A very common structure is $f(\lambda) = 1/(1-\lambda)$ and $g_t(\lambda) = \frac{\gamma}{\lambda}\text{OPT}_t$. For example, consider job scheduling with predictions, a setting where we are given n jobs and their predicted runtimes with total absolute error η and must minimize the sum of their completion times when running on a single server with pre-emption. Here Kumar et al. [24, Theorem 3.3] showed that a preferential round-robin algorithm has competitive ratio at most $\min\left\{\frac{1+2\eta/n}{1-\lambda}, \frac{2}{\lambda}\right\}$. Thus if we know the prediction is perfect we can set $\lambda = 0$ and obtain the optimal cost (consistency); on the other hand, if we know the prediction is poor we can set $\lambda = 1$ and get the (tight) worst-case guarantee of two (robustness).

Of course in-practice we often do not know how good a prediction is on a specific instance t; we thus would like to learn to set λ, i.e. to learn how trustworthy our prediction is. As a first step, we can consider doing so when we are given a prediction for each instance and thus only need to optimize over λ. For example, the just-discussed problem of job scheduling has competitive ratio upper-bounded by $U_t(\lambda) = \min\left\{\frac{1+2\eta_t/n_t}{1-\lambda}, \frac{2}{\lambda}\right\}$ for n_t and η_t the number of jobs and the prediction quality, respectively, on instance t. Assuming a bound B on the average prediction error, we can show that U_t is Lipschitz on $[0, 1]$ and apply the continuous Hedge algorithm of Krichene et al. [23]—an expert algorithm for infinite experts—to get a regret guarantee (proof in A.2):

Corollary 5.1. *For the competitive ratio upper bounds U_t of the job scheduling problem with average prediction error η/n_t at most B there exists an algorithm with expected regret*

$$\max_{\lambda\in[0,1]} \mathbb{E}\sum_{t=1}^{T} U_t(\lambda_t) - U_t(\lambda) \leq 9B\left(1 + \sqrt{\frac{T}{2}\log T}\right)$$

where the expectation is over the randomness of the algorithm.

Thus a standard online Lipschitz learning method is able produce a sequence of parameters λ_t that performs as well as the best tradeoff asymptotically. We next consider the more difficult setting where we wish to learn to predict and tune λ simultaneously.

Ski-rental: We instantiate this challenge on ski-rental, a common setup in which each task t is a ski season with an unknown number of days $n_t \in \mathbb{Z}_{\geq 2}$; to ski each day, we must either buy skis at price b_t or rent each day for the price of one. The optimal offline behavior is to buy iff $b_t < n_t$, and the best algorithm has worst-case competitive ratio $e/(e-1)$. Kumar et al. [24] and Bamas et al. [9, Theorem 2] further derive an algorithm with the following robustness-consistency trade-off between blindly following a prediction x and incurring cost $u_t(x) = b_t 1_{x>b_t} + n_t 1_{x\leq b_t}$ or going with the worst-case guarantee:

$$U_t(x, \lambda) = \frac{\min\{\lambda u_t(x), b_t, n_t\}}{1 - e_t(-\lambda)}, e_t(z) = (1 + 1/b_t)^{b_t z}$$

Assuming a bound $N \geq 2$ on the number of days and $B > 0$ on the buy price we can show that U_t is bounded and Lipschitz w.r.t. λ. We can thus run randomized exponentiated gradient over the product set $[N] \times \{\delta/2, \ldots, 1 - \delta/2\}$ on the functions $U_t(\cdot, x_t)$ for some δ s.t. $1/\delta \in \mathbb{Z}_{\geq 2}$ to get a bound on the expected regret (proof in A.3).

Corollary 5.2. *For the competitive ratio upper bounds U_t of the discrete ski-rental problem there exists an online algorithm with expected regret*

$$\max_{x \in [0,N], \lambda \in (0,1]} \mathbb{E} \sum_{t=1}^{T} U_t(x_t, \lambda_t) - U_t(x, \lambda) \leq 6N\sqrt{T \log(BNT)}$$

Thus via an appropriate discretization the sequence of predictions (x_t, λ_t) does as well as the joint optimum on this problem. However, we can also look at a case where we are not able to just discretize to get low regret. In particular, we consider the *continuous* ski-rental problem, where each day $n_t > 1$ is a real number, and study how to pick thresholds x after which to buy skis, which has cost $u_t(x) = n_t 1_{n_t \leq x} + (b_t + x)1_{n_t > x}$. Note that $x = 0$ and $x = N$ recovers the previous setting where our decision was to buy or not at the beginning. For this setting, Diakonikolas et al. [12] adapt an algorithm of Mahdian et al. [29] to bound the cost as follows:

$$C_t(x, \lambda) \leq U_t(x, \lambda) = \min \left\{ \frac{u_t(x)}{1 - \lambda}, \frac{e \min\{n_t, b_t\}}{(e-1)\lambda} \right\}$$

While the bound is simpler as a function of λ, it is discontinuous in x because u_t is piecewise-Lipschitz. Since one cannot even attain sublinear regret on adversarially chosen threshold functions, we must make an assumption on the data. In particular, we will assume the days are *dispersed*:

Definition 5.3. A set of points $n_1, \ldots, n_T$ is β**-dispersed** for $\beta > 0$ if for all $\varepsilon \geq T^{-\beta}$ the number of points in any ball of radius ε is at most $\tilde{\mathcal{O}}(\varepsilon T)$, i.e. $\max_{x \in [0,N]} |[x \pm \varepsilon] \cap \{n_1, \ldots, n_T\}| = \tilde{\mathcal{O}}(\varepsilon T)$.

Dispersion encodes the stipulation that the days, and thus the discontinuities of $u_t(x, \lambda)$, are not too concentrated. In the i.i.d. setting, a simple condition that leads to dispersion with $\beta = 1/2$ is the assumption that the points are drawn from a κ-bounded distribution [6, Lemma 1]. Notably this is a strictly weaker assumption than the log-concave requirement of Diakonikolas et al. [12] that they used to show statistical learning results for ski-rental. Having stipulated that the ski-days are β-dispersed, we can show that it implies dispersion of the loss functions [6] and thus obtain the following guarantee for the exponentially weighted forecaster [6, 8] applied to $U_t(x, \lambda)$ (proof in A.4):

Corollary 5.4. *For cost upper bounds U_t of the continuous ski-rental problem there exists an online algorithm with expected regret*

$$\max_{x \in [0,N], \lambda \in (0,1]} \mathbb{E} \sum_{t=1}^{T} U_t(x_t, \lambda_t) - U_t(x, \lambda) \leq \tilde{\mathcal{O}}\left(\sqrt{T \log(NT)} + (N + B)^2 T^{1-\beta}\right)$$

Thus in two mathematically quite different settings of ski-rental we can directly apply online learning to existing bounds to not only learn online the best action for ski-rental, but to at the same time learn how trustworthy the best action is via tuning the robustness-consistency trade-off.

6 Conclusion and future work

The field of algorithms with predictions has been successful in circumventing worst case lower bounds and showing how simple predictions can improve algorithm performance. However, except for a few problem-specific approaches, the question of *how* to predict has largely been missing from the discussion. In this work we presented the first general framework for efficiently learning useful predictions and applied it to a diverse set of previously studied problems, giving the first low regret learning algorithms, reducing sample complexity bounds, and showing how to learn the best consistency-robustness trade-off. One current limitation is the lack of more general-case guarantees for *simultaneously* tuning robustness-consistency and learning the predictor, which we only show for ski-rental. There are also several other avenues for future work. The first is to build on our results and provide learning guarantees for other problems where the algorithmic question of *how* to use predictions is already addressed. Another is to try to improve known bounds by solving the problems holistically: developing easy-to-learn parameters in concert with developing algorithms that can use them. Finally, there is the direction of identifying hard problems: what are the instances where no reasonable prediction can help improve an algorithm's performance?

References

[1] Keerti Anand, Rong Ge, and Debmalya Panigrahi. Customizing ML predictions for online algorithms. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[2] Keerti Anand, Rong Ge, Amit Kumar, and Debmalya Panigrahi. A regression approach to learning-augmented online algorithms. In *Advances in Neural Information Processing Systems*, 2021.

[3] Maria-Florina Balcan. Data-driven algorithm design. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, Cambridge, UK, 2021.

[4] Maria-Florina Balcan and Avrim Blum. Approximation algorithms and online mechanisms for item pricing. *Theory of Computing*, 3:179–195, 2007.

[5] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[6] Maria-Florina Balcan, Travis Dick, and Ellen Vitercik. Dispersion for data-driven algorithm design, online learning, and private optimization. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 603–614, 2018.

[7] Maria-Florina Balcan, Travis Dick, and Wesley Pegden. Semi-bandit optimization in the dispersed setting. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2020.

[8] Maria-Florina Balcan, Mikhail Khodak, Dravyansh Sharma, and Ameet Talwalkar. Learning-to-learn non-convex piecewise-Lipschitz functions. In *Advances in Neural Information Processing Systems*, 2021.

[9] Etienne Bamas, Andreas Maggiori, and Ola Svensson. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems*, 2020.

[10] Nicolò Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

[11] Ofer Dekel, Arthur Flajolet, Nika Haghtalab, and Patrick Jaillet. Online learning with a hint. In *Advances in Neural Information Processing Systems*, 2017.

[12] Ilias Diakonikolas, Vasilis Kontonis, Christos Tzamos, Ali Vakilian, and Nikos Zarifis. Learning online algorithms with distributional advice. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[13] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Faster matchings via learned duals. In *Advances in Neural Information Processing Systems*, 2021.

[14] Elbert Du, Franklyn Wang, and Michael Mitzenmacher. Putting the "learning" into learning-augmented algorithms for frequency estimation. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[16] Rishi Gupta and Timothy Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[17] Piotr Indyk, Frederik Mallmann-Trenn, Slobodan Mitrović, and Ronitt Rubinfeld. Online page migration with ML advice. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, 2022.

[18] Ali Jadbabaie, Alexander Rakhlin, and Shahin Shahrampour. Online optimization: Competing with dynamic comparators. In *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics*, 2015.

[19] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. Online algorithms for weighted paging with predictions. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming*, 2020.

[20] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Adaptive gradient-based meta-learning methods. In *Advances in Neural Information Processing Systems*, 2019.

[21] Mikhail Khodak, Renbo Tu, Tian Li, Liam Li, Maria-Florina Balcan, Virginia Smith, and Ameet Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. In *Advances in Neural Information Processing Systems*, 2021.

[22] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data*, 2018.

[23] Walid Krichene, Maximilian Balandat, Claire Tomlin, and Alexandre Bayen. The hedge algorithm on a continuum. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[24] Ravi Kumar, Manish Purohit, and Zoya Svitkina. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, 2018.

[25] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, 2020.

[26] Jeffrey Li, Mikhail Khodak, Sebastian Caldas, and Ameet Talwalkar. Differentially private meta-learning. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

[27] Sen Lin, Mehmet Dedeoglu, and Junshan Zhang. Accelerating distributed online meta-learning via multi-agent collaboration under limited communication. In *Proceedings of the Twenty-second International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2021.

[28] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. 68(4), 2021.

[29] Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Transactions on Algorithms*, 8:1–29, 2012.

[30] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, Cambridge, UK, 2021.

[31] Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *Proceedings of the 26th Annual Conference on Learning Theory*, 2013.

[32] Alexander Rakhlin and Karthik Sridharan. Efficient online multiclass prediction on graphs via surrogate losses. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[33] Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, 2020.

[34] Timothy Roughgarden. *Beyond Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[35] Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. Uniform bounds for scheduling with job size estimates. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference*, 2022.

[36] Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

[37] Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *Journal of the ACM*, 51(3):385–463, 2004.

[38] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 2003.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Assumptions stated in Sections 3, 4, 5, and the Appendix.
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs given in Sections 3, 4, and the Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [N/A]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [N/A]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [N/A]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs of main results

A.1 Proof of Lemma 4.1

Proof. For each $j \in [n]$ define $p_j = 1 - \langle \mathbf{s}_{[j]}, \mathbf{p}_{[j]} \rangle$, i.e. the probability that $\hat{s}_j \neq s_j$, and the r.v. $X_j \sim \mathrm{Ber}(p_j)$. Define also the r.v. $S_i = \sum_{j=i}^{i+\gamma D-1} X_j$, s.t. we have

$$\gamma D \mathbb{E}_{\mathbf{p}} q = \mathbb{E}_{\mathbf{p}} \max_{i \in [n-\gamma D+1]} S_i = \mathbb{E}_{\mathbf{p}} \max_{i \in [n-\gamma D+1]} \sum_{j=i}^{i+\gamma D-1} X_j$$

Note that S_i is a Poisson binomial and so has moment-generating function $\mathbb{E}_{\mathbf{p}} \exp(tS_i) = \prod_{j=i}^{i+\gamma D-1}(1 - p_j + p_j e^t)$. Therefore applying Jensen's inequality and the union bound yields

$$\begin{aligned}
\exp\left(t \mathbb{E}_{\mathbf{p}} \max_{i \in [n-\gamma D+1]} S_i\right) \leq \mathbb{E}_{\mathbf{p}} \exp\left(t \max_{i \in [n-\gamma D+1]} S_i\right) &= \mathbb{E}_{\mathbf{p}} \max_{i \in [n-\gamma D+1]} \exp\left(tS_i\right) \\
&\leq \sum_{i=1}^{n-\gamma D+1} \mathbb{E}_{\mathbf{p}} \exp\left(tS_i\right) \\
&\leq (n - \gamma D + 1) \prod_{j=i^*}^{i^*+\gamma D-1} (1 - p_j + p_j e^t)
\end{aligned}$$

for all $t > 0$ and $i^* \in \arg\max_{i \in [n-\gamma D+1]} \mathbb{E}_{\mathbf{p}} S_i$. We then have

$$\begin{aligned}
t \mathbb{E}_{\mathbf{p}} \max_{i \in [n-\gamma D+1]} S_i &\leq \log(n - \gamma D + 1) + \sum_{j=i^*}^{i^*+\gamma D-1} \log(1 - p_j + p_j e^t) \\
&\leq \log(n - \gamma D + 1) + \sum_{j=i^*}^{i^*+\gamma D-1} \log \exp(p_j(e^t - 1)) \\
&\leq \log(n - \gamma D + 1) + \sum_{j=i^*}^{i^*+\gamma D-1} p_j(e^t - 1) \\
&\leq \log(n - \gamma D + 1) + \mathbb{E}_{\mathbf{p}} S_{i^*}(e^t - 1)
\end{aligned}$$

Dividing by $t = W\left(\frac{\log(n-\gamma D+1)}{x}\right) + 1$ shows that $f(x) = \frac{x(\exp(t)-1)+\log(n-\gamma D+1)}{t\gamma D}$, where $W : [0, \infty) \mapsto [0, \infty)$ is the Lambert W-function. Define $L = \log(n - \gamma D + 1)/e$, so we are interested in bounding $f(x) = \frac{x(\exp(W(L/x)+1)-1)+eL}{W(L/x)+1}$. We compute its derivative:

$$f'(x) = \frac{x(\exp(W(L/x) + 1) - 1)W(L/x)^2 - 3x(W(L/x) + 1/3) + x\exp(W(L/x) + 1) + 2eL}{x(W(L/x) + 1)^3}$$

and second derivative:

$$f''(x) = -\frac{W(L/x)\left((x + eL)W(L/x)^2 + 2(2x + eL)W(L/x) + eL\right)}{x^2(W(L/x) + 1)^5}$$

Since the second derivative is always negative, f is a concave function on $x \geq 0$. Thus for $\omega = W(1)$ we have

$$\begin{aligned}
f(x) &\leq \min_{y>0} f(y) + f'(y)(x - y) \\
&\leq \frac{L(e/\omega - 1 + e)}{\omega + 1} + \frac{L(e/\omega - 1)\omega^2 - 3L(\omega + 1/3) + Le/\omega + 2eL}{L(\omega + 1)^3}(x - L) \\
&= \left(e/\omega + 1/(\omega + 1)^3 - (e + 1)/(\omega + 1) - 1/(\omega + 1)^2\right) x \\
&\quad + \left(1/(\omega + 1)^2 + e/(\omega + 1) - 1/(\omega + 1)^3\right) L \\
&< ex + \frac{2}{e}\log(n - \gamma D + 1)
\end{aligned}$$

□

A.2 Proof of Corollary 5.1

Proof. We have that $U_t(\lambda)$ is bounded above by $3(1+2B)$, its largest gradient is attained at $2/(3+2\eta_t/n)$ where it is bounded by $(3+2B)/2$. Applying Krichene et al. [23, Corollary 2] and simplifying yields the result. □

A.3 Proof of Corollary 5.2

Proof. $U_t(x,\lambda)$ is bounded above by $2N$ and its largest gradient is attained at $\lambda = \frac{\min\{b_t,n_t\}}{u_t(x)} \geq \frac{1}{N}$ with norm bounded by $\frac{B\exp(1/N)}{(\exp(1/N)-1)^2}$. Let $\Lambda = \{k\delta\}_{k=1}^{\lfloor 1/\delta \rfloor}$ for some $\delta \in (0,1]$. Then we run EG on the simplex over $[N]\times\Lambda$ and with step-size $\frac{1}{2N}\sqrt{\frac{\log\frac{N}{\delta}}{2T}}$ to obtain regret compared to the best element of $[N]\times\Lambda$ of $2N\sqrt{2T\log\frac{N}{\delta}}$ [36, Theorem 2.15]. Setting $\delta = \min\left\{\frac{N(\exp(1/N)-1)^2}{B\exp(1/N)}\sqrt{\frac{2}{T}}, 1\right\}$ yields

$$\begin{aligned}
\mathbb{E}\sum_{t=1}^{T} U_t(x_t,\lambda_t) &\leq 2N\sqrt{2T\log(N\lfloor 1/\delta\rfloor)} + \min_{x\in[N],\lambda\in\Lambda}\sum_{t=1}^{T} U_t(x,\lambda) \\
&\leq 2N\sqrt{2T\log\frac{N}{\delta}} + \frac{B\exp(1/N)\delta T}{(\exp(1/N)-1)^2} + \min_{x\in[N],\lambda\in(0,1]}\sum_{t=1}^{T} U_t(x,\lambda) \\
&\leq 2N\sqrt{2T\left(\log(BT) + \max\left\{\log N, \frac{1}{N} - 2\log\left(\exp\left(\frac{1}{N}\right)-1\right)\right\}\right)} \\
&\quad + N\sqrt{2T} + \min_{x\in[N],\lambda\in(0,1]}\sum_{t=1}^{T} U_t(x,\lambda) \\
&\leq 6N\sqrt{T\log(BNT)} + \min_{x\in[N],\lambda\in(0,1]}\sum_{t=1}^{T} U_t(x,\lambda)
\end{aligned}$$

□

A.4 Proof of Corollary 5.4

Proof. $U_t(x,\lambda)$ is bounded above by $e(N+B)$, its largest gradient w.r.t. λ is attained at $\lambda = \frac{e\min\{n_t,b_t\}}{(e-1)u_t(x)+e\min\{n_t,b_t\}}$, where it is bounded by $\left(\frac{2e(N+B)}{e-1}\right)^2$, and its largest gradient w.r.t. x is $e(N+B)$. Thus the function is $5e(N+B)^2$-Lipschitz w.r.t. the 2-norm, apart from discontinuities at $x = n_t$. Now, note that our assumption that the points $n_1,\ldots,n_T$ are β-dispersed implies exactly that the functions U_t are β-dispersed (c.f. Balcan et al. [8, Definition 2.1]), so the exponentially-weighted forecaster attains expected regret $\tilde{\mathcal{O}}\left(\sqrt{T\log(NT)} + (N+B)^2T^{1-\beta}\right)$. □

B b-Matching

Definition B.1. For $\mathbf{b} \in \mathbb{R}^n_{\geq 0}$ the **b-seminorm** $\|\cdot\|_{\mathbf{b},1} : \mathbb{R}^n \mapsto \mathbb{R}_{\geq 0}$ is $\|\mathbf{x}\|_{\mathbf{b},1} = \sum_{i=1}^n \mathbf{b}_{[i]}|\mathbf{x}_{[i]}|$.

Claim B.1. *Given any vectors $\mathbf{x} \in \mathbb{Z}^n$ and $\mathbf{y} \in \mathbb{R}^n$, let $\tilde{\mathbf{y}} \in \mathbb{Z}^n$ be the vector whose elements are those of $\mathbf{y}$ rounded to the nearest integer. Then for all $\mathbf{b} \in \mathbb{Z}^n$ we have $\|\mathbf{x} - \tilde{\mathbf{y}}\|_{\mathbf{b},1} \leq 2\|\mathbf{x} - \mathbf{y}\|_{\mathbf{b},1}$.*

Proof. Let $S \subset [n]$ be the set of indices $i \in [n]$ for which $\mathbf{x}_{[i]} \geq \mathbf{y}_{[i]} \iff \tilde{\mathbf{y}}_{[i]} = \lceil \mathbf{y}_{[i]} \rceil$. For $i \in [n] \backslash S$ we have $|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| \geq 1/2 \geq |\tilde{\mathbf{y}}_{[i]} - \mathbf{y}_{[i]}|$ so it follows by the triangle inequality that

$$\begin{aligned}
\|\mathbf{x} - \tilde{\mathbf{y}}\|_{\mathbf{b},1} &= \sum_{i \in S} \mathbf{b}_{[i]}|\mathbf{x}_{[i]} - \tilde{\mathbf{y}}_{[i]}| + \sum_{i \in [n]\backslash S} \mathbf{b}_{[i]}|\mathbf{x}_{[i]} - \tilde{\mathbf{y}}_{[i]}| \\
&\leq \sum_{i \in S} \mathbf{b}_{[i]}|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + \sum_{i \in [n]\backslash S} \mathbf{b}_{[i]}(|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + |\mathbf{y}_{[i]} - \tilde{\mathbf{y}}_{[i]}|) \\
&\leq \sum_{i \in S} \mathbf{b}_{[i]}|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| + 2 \sum_{i \in [n]\backslash S} \mathbf{b}_{[i]}|\mathbf{x}_{[i]} - \mathbf{y}_{[i]}| \leq 2\|\mathbf{x} - \mathbf{y}\|_{\mathbf{b},1}
\end{aligned}$$

□

Theorem B.2. *Suppose we have a fixed graph with n vertices and m edges.*

1. *For any cost vector $\mathbf{c} \in \mathbb{Z}^m_{\geq 0}$, any demand vector $\mathbf{b} \in \mathbb{Z}^n_{\geq 0}$, and any dual vector $\hat{\mathbf{x}} \in \mathbb{R}^n$ there exists an algorithm for minimum weight perfect $\mathbf{b}$-matching that runs in time $\tilde{\mathcal{O}}\left(mnU(\hat{\mathbf{x}})\right)$, where $U(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1}$ for $\mathbf{x}^*(\mathbf{c}, \mathbf{b})$ the optimal dual vector associated with $\mathbf{c}$ and $\mathbf{b}$.*
2. *For any $\delta, \varepsilon > 0$ and any distribution $\mathcal{D}$ over (cost, demand) vector pairs in $\mathbb{Z}^m_{\geq 0} \times \mathbb{Z}^n_{\geq 0}$ with respective ℓ_∞-norms bounded by C and B there exists a poly-time algorithm that takes $\mathcal{O}\left(\left(\frac{nCB}{\varepsilon}\right)^2 \left(n + \log \frac{1}{\delta}\right)\right)$ samples from $\mathcal{D}$ and returns $\hat{\mathbf{x}}$ s.t. w.p. $\geq 1 - \delta$:*
$$\mathbb{E}_{(\mathbf{c},\mathbf{b})\sim\mathcal{D}}\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1} \leq \min_{\|\mathbf{x}\|_\infty \leq C} \mathbb{E}_{(\mathbf{c},\mathbf{b})\sim\mathcal{D}}\|\mathbf{x} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1} + \varepsilon$$
3. *Let $(\mathbf{c}_1, \mathbf{b}_1), \ldots, (\mathbf{c}_T, \mathbf{b}_T) \in \mathbb{Z}^m_{\geq 0} \times \mathbb{Z}^n_{\geq 0}$ be an adversarial sequence of (cost, demand) vector pairs with ℓ_∞-norms bounded by C and B, respectively. There exists an efficient algorithm that predicts $\hat{\mathbf{x}}_t$ after seeing $(\mathbf{c}_1, \mathbf{b}_1), \ldots, (\mathbf{c}_{t-1}, \mathbf{b}_{t-1})$ and achieves regret*
$$\max_{\|\mathbf{x}\|_\infty \leq C} \sum_{t=1}^T \|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c}_t, \mathbf{b}_t)\|_{\mathbf{b}_t,1} - \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t, \mathbf{b}_t)\|_{\mathbf{b}_t,1} \leq CBn\sqrt{2nT}$$

Proof. The first result follows by combining Dinitz et al. [13, Theorem 31] with Claim B.1. For the third, let $\hat{\mathbf{x}}_t$ be the sequence generated by running OGD [38] with step size $\frac{C}{B\sqrt{2nT}}$ on the losses $U_t(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t, \mathbf{b}_t)\|_{\mathbf{b}_t,1}$ over domain $[-C, C]^n$. Since these losses are Bn-Lipschitz and the duals are $C\sqrt{n}$-bounded in Euclidean norm the regret guarantee follows from Shalev-Shwartz [36, Corollary 2.7]. For the second result, we apply standard online-to-batch conversion to the third result, i.e. we draw $T = \Omega\left(\left(\frac{nCB}{\varepsilon}\right)^2 \left(n + \log \frac{1}{\delta}\right)\right)$ samples $(\mathbf{c}_t, \mathbf{b}_t)$, run OGD on the resulting losses $U_t(\mathbf{x})$, and set $\hat{\mathbf{x}} = \frac{1}{T}\sum_{t=1}^T \hat{\mathbf{x}}_t$ to be the average of the resulting predictions $\hat{\mathbf{x}}_t$. Then applying Jensen's inequality, Cesa-Bianchi et al. [10, Proposition 1], the regret guarantee, and Hoeffding's inequality yields

$$\begin{aligned}
\mathbb{E}_{\mathbf{c},\mathbf{b}}\|\hat{\mathbf{x}} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1} &\leq \frac{1}{T}\sum_{t=1}^T \mathbb{E}_{\mathbf{c},\mathbf{b}}\|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1} \\
&\leq \frac{1}{T}\sum_{t=1}^T \|\hat{\mathbf{x}}_t - \mathbf{x}^*(\mathbf{c}_t, \mathbf{b}_t)\|_{\mathbf{b}_t,1} + CBn\sqrt{\frac{2}{T}\log\frac{2}{\delta}} \\
&\leq \min_{\|\mathbf{x}\|_\infty \leq C} \frac{1}{T}\sum_{t=1}^T \|\mathbf{x} - \mathbf{x}^*(\mathbf{c}_t, \mathbf{b}_t)\|_{\mathbf{b}_t,1} + \frac{CBn\sqrt{2n} + CBn\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}} \\
&\leq \min_{\|\mathbf{x}\|_\infty \leq C} \mathbb{E}_{\mathbf{c},\mathbf{b}}\|\mathbf{x} - \mathbf{x}^*(\mathbf{c}, \mathbf{b})\|_{\mathbf{b},1} + \frac{CBn\sqrt{2n} + 2CBn\sqrt{2\log\frac{2}{\delta}}}{\sqrt{T}}
\end{aligned}$$

□