Learning Predictions for Algorithms with Predictions

Misha Khodak, Nina Balcan, Ameet Talwalkar, Sergei Vassilvitskii
khodak@cmu.edu

Algorithms with predictions

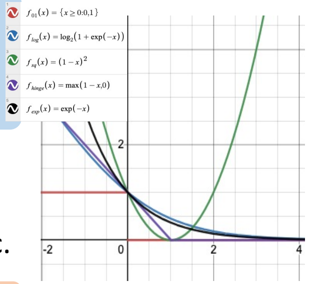
take advantage of a prediction w to improve the cost $C_x(w)$ of running on an instance x

generic guarantee: $C_x(w)$ is bounded by a function $U_x(w)$, which is

1. small if prediction w is good (**consistency**)
2. as good as the worst-case (**robustness**)

A new framework for **learning predictions**

Step 1: derive a "nice" upper bound U_x of C_x



- U_x should be a **surrogate** loss for C_x
- convex, Lipschitz, etc.

Step 2: apply **online learning**

Results:

- low regret: $\sum_t U_{x_t}(w_t) - \min_w \sum_t U_{x_t}(w) = o(T)$
- sample complexity: $\mathbb{E}_x U_x(\hat{w}) \leq \min_w \mathbb{E}_x U_x(w) + \varepsilon$
- instance-dependent prediction: $w \leftarrow \langle \mathbf{a}, \mathbf{f}_x \rangle$
- problem-specific learning algorithms

Standard approach

derive upper bound $U_x(w)$ on the cost of the algorithm → no learning guarantee ☹

bound sample complexity of via VC/pseudo-dimension, etc. → **Results**: - sample complexity bound

what algorithm designers are already good at

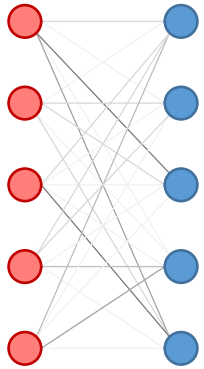
Our framework

derive upper bound $U_x(w)$ on the cost of the algorithm → make the upper bound nice → apply online learning →

Results:
- low regret guarantee
- sample complexity bound
- instance-dependent predictors
- practical and efficient algorithms

Example: Bipartite matching

Problem: for a bipartite graph with m edges and n vertices, find the perfect matching with least weight according to edge-weights $\boldsymbol{x} \in \mathbb{Z}^m$



Algorithm [1]: Hungarian method initialized at **integer** duals $\boldsymbol{w} \in \mathbb{Z}^n$ has runtime $O(m\sqrt{n}\|\boldsymbol{w} - \boldsymbol{y}^*(\boldsymbol{x})\|_1)$, where $\boldsymbol{y}^*(\boldsymbol{x}) \in \mathbb{Z}^n$ is the dual vector of the optimum

Step 1: rounding $\boldsymbol{w} \in \mathbb{R}^n$ to the integers before running Hungarian

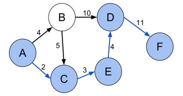
- preserves distance to $\boldsymbol{y}^*(\boldsymbol{c})$ up to constants
- makes the problem of learning $\boldsymbol{w}$ convex

Step 2: apply online gradient descent to $U_{\boldsymbol{x}}(\boldsymbol{w}) = \|\boldsymbol{w} - \boldsymbol{y}^*(\boldsymbol{x})\|_1$

- $\tilde{\Omega}(n^2/\varepsilon^2)$ samples needed to PAC-learn $\boldsymbol{w}$ ← $\tilde{O}(n)$ improvement over [1]!
- $O(n\sqrt{2T})$ cumulative regret ← first regret guarantee

More applications

Better bounds for shortest path and $\boldsymbol{b}$-matching



We extend our matching guarantees to obtain up to $O(n^2)$ improvement in sample complexity over [2]

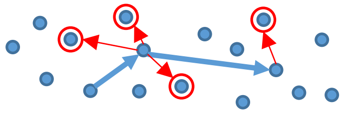
First learning guarantees for online page migration

Goal: predict a distribution over a finite metric space K to satisfy a sequence of n requests

Step 1: make existing guarantee [3] convex at cost $O(\log n)$

Step 2: apply exponentiated gradient descent

- regret: $O(n\sqrt{T \log|K|})$
- sample complexity: $O\left(\frac{n^2}{\varepsilon^2}\log|K|\right)$



Tuning robustness-consistency tradeoffs

Robustness-consistency can be traded off parametrically:

$$C_x(w, \lambda) \leq \min\{f(\lambda)U_x(w), g_x(\lambda)\}$$

for f increasing, g decreasing, and $\lambda \in [0,1]$.

We show how to learn the best λ using data, sometimes simultaneously with learning the prediction.

Learning predictions for job scheduling

See paper (arxiv.org/abs/2202.09312) for learning predictions

- that improve the fractional makespan in online scheduling
- of optimal job permutations for non-clairvoyant scheduling

References:

[1] Dinitz, Im, Lavastida, Moseley, Vassilvitskii, NeurIPS 2021
[2] Chen, Silwal, Vakilian, Zhang. ICML 2022.
[3] Indyk, Mallmann-Trenn, Mitrović, Rubinfeld, AISTATS 2022