Online Algorithms with Multiple Predictions

Keerti Anand[1], Rong Ge[1], Amit Kumar[2], Debmalya Panigrahi[1]

[1]Department of Computer Science, Duke University
[2]Department of Computer Science, Indian Institute of Technology Delhi

Introduction

Online Algorithms with Predictions: An attempt to bypass the pessimistic worst-case bounds of traditional algorithm design.

Design an algorithm whose performance improves with the accuracy of the prediction; yet maintaining an inherent worst-case guarantee. Example Problems : Ski Rental, Scheduling, Caching, Matching and Secretary Problems.

Algorithms with Multiple Predictions

Multiple Predictions Setting: Instead of one prediction, what if we get multiple sets of predictions?

Multiple ML algorithms, human experts can suggest their suitable solutions.

Some solutions can be arbitrarily bad but as long as one solution is good, our algorithm should be able to find it.

Goal: Online algorithm whose performance is competitive against that of the *best* predictor

Online Covering Framework

$$min \sum_{i=1}^{n} c_i \cdot x_i$$

Objective

$$\sum_{i=1}^{n} a_{ij} \cdot x_i \geq 1$$

At each step j:
New covering constraint arrives!

Applications

The Online Covering Framework (OCF) can be applied to online versions of a variety of problems such as:

1. Set Cover
2. Caching
3. Facility Location*

* Uses an extension of online covering framework that has box-constraints

Online Covering with Multiple Predictions

j-th covering constraint arrives! $\sum_{i=1}^{n} a_{ij} \cdot x_i \geq 1$

We get k sets of suggestions: $x_i \longrightarrow x_i(j, s)$

Benchmark:

- An optimal solution that chooses the same suggestion in each time step. *STATIC*
- An optimal solution that chooses any of the suggestions in each time step *DYNAMIC*

Our Results

Theorem 2.1. *There is an algorithm for the online covering problem with k suggestions that has a competitive ratio of $O(\log k)$, even against the* DYNAMIC *benchmark.*

Theorem 2.2. *The competitive ratio of any algorithm for the online covering problem with k suggestions is $\Omega(\log k)$, even against the* STATIC *benchmark.*

Algorithm Intuition

The rate at which a variable is increased depends on three key factors:

1. How strongly a variable is suggested?

If a variable shows up in multiple suggestions, we should increase it faster!

2. How costly the variable is?

If a variable is costly, then we should increase it slowly!

3. How much it contributes to the covering constraint?

Higher the contribution to the covering constraint, faster we should increase the variable

Algorithm 1: Online Covering Algorithm

1.1 **Offline:** All variables x_i are initialized to 0.

1.2 **Online:** On arrival of the j-th constraint:

1.3 **while** $\sum_{i=1}^{n} a_{ij} x_i < \frac{1}{2}$,

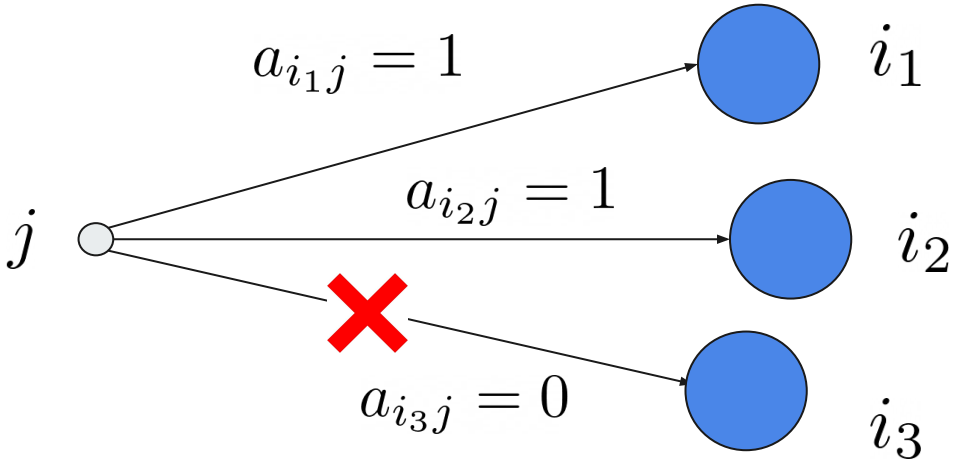
1.4 **for** $i \in [n]$

1.5 **if** $x_i < \frac{1}{2}$, increase x_i by $\boxed{\frac{dx_i}{dt} = \frac{a_{ij}}{c_i}\left(x_i + \delta \cdot x_{ij}\right)}$,

1.6 where $\delta = \frac{1}{k}$ and $x_{ij} = \sum_{s=1}^{k} x_i(j, s)$.

Online Set Cover

At each time, element j arrives and we must have enough sets in our solution to cover it



$$min \sum_{i=1}^{n} c_i \cdot x_i$$

$$\sum_{i=1}^{n} a_{ij} \cdot x_i \geq 1$$

Robustification

Theorem 2.3. *Suppose a class of online covering problems have an online algorithm (without predictions) whose competitive ratio is α. Then, there is an algorithm for this class of online covering problems with k suggestions that produces an online solution whose cost is at most* $O(\min\{\log k \cdot \text{DYNAMIC}, \alpha \cdot opt\})$.

Conclusion

We give a general recipe to design online algorithms for covering with multiple machine-learned predictions.

Future Extensions:

Packing problems such as budgeted allocation.

Covering problems with non-linear (convex) objectives.