Improving Online Algorithms via ML Predictions

Ravi Kumar, Manish Purohit, Zoya Svitkina
NeurIPS2018

Juan Vanerio

Communication Technologies Group
Faculty of Computer Science
University of Vienna

November 22, 2021

Table of Contents

Introduction

Introduction

Introduction

- **Online Algorithms (OA)**
- Algorithms that make decisions one request at a time.
- Serve an sequence of item requests σ.
- Usually Oblivious adversary model generates arbitrary sequence.
- Online: No knowledge of future requests.
  - Given a time index t ordering the sequence, knowledge is available $\forall s \leq t$
- Research focus on *competitive analysis*.

$$ALG(\sigma) \leq c \cdot OPT(\sigma) + \alpha$$

- **Empirical** performance is **usually better** than suggested by competitive analysis.

Introduction

- OA are designed considering worst-case scenarios.
- **Can we improve the quality of OA if we have more information?**
  - Implication: weaker adversary.
- It is possible if predictions are available.
  - Reduced uncertainty.
- Competitive analysis no longer directly applies $\Rightarrow$ New framework.
- The paper explains through examples a method of extending OA whit predictions.
  - Skii rental.
  - Non-clairvoyant job scheduling.

Introduction - What to expect

What is the paper about:

- Examples on a method for extending online algorithms when predictions are available.
- A framework for its analysis.

What is the paper **not** about:

- The list update problem.
- Any particular cost model.
- Machine Learning – Predictions instead, overlap but not necessarily the same.

Framework

Framework

Predictions

- Uncertainty is addressed by making predictions.
- Assume predictor P s.t. makes an estimation y of a yet unknown value x.
- Let $\eta = \eta(x, y)$ be the **predictor error**.
  - **Problem dependent**.
  - $\eta \geq 0$.
  - If $x = y$ then $\eta = 0$.
- What if the error is large?
- If predictions are used naively, the end result could be arbitrarily bad!

Concepts

- Traditional algorithms: Competitive analysis, assumes worst-case scenarios.
  - Let c be the competitive ratio $c = \frac{ALG(\sigma)}{OPT(\sigma)}$
- Idea: express c as a function of predictor error η: $c(\eta)$.

Definition: γ-robustness

ALG is γ-robust if $\exists \gamma$ s.t. $\forall \eta$, $c(\eta) \leq \gamma$

- Measures algorithm performance under bad predictions.
- Smaller γ is better.

Definition: β-consistency

ALG is β-consistent if $c(0) = \beta$

- How well the algorithm does with perfect (errorless) predictions.
- Smaller β is better.

Design

Key design idea:

- **Hyperparameter**: Introduce an hyperparameter $\lambda \in (0, 1)$ into the online algorithms.
- λ: **"distrust in the prediction"**
  - λ near 0: Complete trust in the prediction.
  - λ near 1: Complete distrust in the prediction.
- **Intuition: Control with λ the impact of considering the prediction on the algorithm.**
- Trade-off between robustness and consistency.

Skii rental

Skii Rental

Skii rental

- Problem:
  - A skier can rent skiis at cost 1 per day or buy them at cost $b > 1$ and skii for free from then on.
  - Its objective is to minimize the cost.
  - The uncertainty is on the number of skiing days x.
  - Predicted number of skiing days: y.
  - Prediction error: $\eta = |x - y|$
- Optimal deterministic OA: **break-even**:
  - Rent on the first $b - 1$ days, buy on day b.
  - 2-competitive (tight)
- There is also a $\frac{e}{1-e}$ randomized OA.

Simple consistent non-robust algorithm

Algorithm 1 A simple 1-consistent algorithm

1: **if** $y \geq b$ **then**
2: Buy at the first day.
3: **else**
4: Keep renting for all the skiing days.
5: **end if**

Proof:

- $y \geq b, x \geq b \Rightarrow ALG = b = OPT$
- $y < b, x < b \Rightarrow ALG = x = OPT$
- $y \geq b, x < b \Rightarrow ALG = b \leq x + y - x = x + \eta = OPT + \eta$
- $y < b, x \geq b \Rightarrow ALG = x < b + x - y = b + \eta = OPT + \eta$ □.
- ∞-robust (bad!)

Deterministic algorithm

Algorithm 2 A deterministic robust and consistent algorithm

1: **if** $y \geq b$ **then**
2: Buy at the start of day $\lceil \lambda b \rceil$.
3: **else**
4: Buy at the start of day $\lceil b/\lambda \rceil$.
5: **end if**

- Competitive ratio: $\min\{\frac{1+\lambda}{\lambda}, (1+\lambda) + \frac{\eta}{(1-\lambda)OPT}\}$.
- $(1 + 1/\lambda)$-robust.
- $(1 + \lambda)$-consistent.

Analysis –Proof outline

Case 1

- If $y \geq b$ then buy the skis on day $\lceil \lambda b \rceil < b$.
  - If $x \geq \lceil \lambda b \rceil$ then $ALG = b + \lceil \lambda b \rceil - 1 \leq \left(\frac{1+\lambda}{\lambda} \right) \lambda b$, $OPT \geq \lceil \lambda b \rceil \Rightarrow c \leq \frac{1+\lambda}{\lambda}$

- If $y < b$ then buy on day $\lceil b/\lambda \rceil$. Worst case ratio happens if $x = \lceil b/\lambda \rceil$.
  - $ALG = b + \lceil \lambda b \rceil - 1 \leq \left(\frac{1+\lambda}{\lambda} \right) \lambda b$
  - $OPT = b \Rightarrow c \leq \frac{1+\lambda}{\lambda}$

Analysis –Proof outline

Case 2

If $y \geq b$ then buy the skis on day $\lceil \lambda b \rceil < b$,

- $\forall x < \lceil \lambda b \rceil \Rightarrow ALG = OPT = x$
- $\forall x \geq \lceil \lambda b \rceil \Rightarrow ALG = b + \lceil \lambda b \rceil - 1 \leq (1+\lambda)b \leq (1+\lambda)(OPT + \eta)$

If $y < b$ then buy the skis on day $\lceil \lambda b \rceil < b$,

- $\forall x < b \Rightarrow ALG = OPT = x$
- $\forall x \in (b, \lceil b/\lambda \rceil) \Rightarrow ALG = x \leq y + \eta < b + \eta = OPT + \eta$
- $\forall x \geq \lceil b/\lambda \rceil$
  - $\eta = x - y > b/\lambda - b = (1-\lambda)b/\lambda$
  - $\Rightarrow ALG = b + \lceil b/\lambda \rceil - 1 \leq b(1 + 1/\lambda) < b + (\frac{1}{1-\lambda})\eta = OPT + (\frac{1}{1-\lambda})\eta$

$$\Rightarrow ALG \leq (1+\lambda)OPT + (\frac{1}{1-\lambda})\eta$$

Randomized algorithm

Algorithm 3 A randomized robust and consistent algorithm

1: **if** $y \geq b$ **then**
2: Let $k \leftarrow \lfloor \lambda b \rfloor$
3: **else**
4: Let $k \leftarrow \lceil b/\lambda \rceil$
5: **end if**
6: Define distribution q s.t. $\forall 1 \leq i \leq k$, $q_i \propto (1 - 1/b)^{-i}$.
7: Buy at the start of day j sampled from q.

- Competitive ratio: $\min\{\frac{1}{1 - e^{-(\lambda - 1/b)}}, \frac{\lambda}{(1 - e^{-\lambda})}(1 + \frac{\eta}{OPT})\}$.
- $(\frac{1}{1 - e^{-(\lambda - 1/b)}})$-robust.
- $(\frac{\lambda}{1 - e^{-\lambda}})$-consistent.

Analysis –proof outline.

- A case by case study again, very similar to proof of deterministic case.
- Expectations are computed directly on the distributions q.
- Exponentials appearing from the sums are upper bounded with natural base exponentials.

Randomization helps

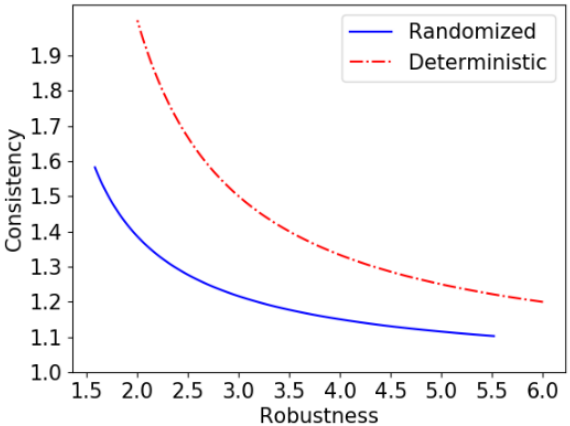


Figure: figure 1 from main paper.

Non-clairvoyant job scheduling with prediction

Non-clairvoyant job scheduling

- Problem:
  - Schedule n jobs on a single machine with no release date.
  - Jobs' length is at least 1.
  - Processing required time x_j of job j is unknown until finished.
  - Any job can be preempted at any time and resumed later at no cost.
  - Objective: minimize sum of flow competition times (*Makespan*).
  - Prediction error: $\eta = \sum_{j=1}^{n} |x_j - y_j|$
- Clairvoyant optimal algorithm: **Shortest Remaining Job First (SRJF)**
  - On any new arrival start processing the job with the least remaining processing time.
  - 1-competitive.
- Optimal algorithm in the non-clairvoyant case: **Round-Robin (RR)**
  - All k unfinished jobs run at the machine at equal rates of $1/k$.
  - 2-competitive (tight).

Algorithm with predictions – design

- We have predictions available.
- RR gets no benefit from predictions.
- Our worst case should then be like RR (2-comp).
- If predictions are perfect we are in the clairvoyant case!
  - We should be then able to perform as good as SRJF.
- Enter **Shortest Predicted Job First (SPJF)**.
  - Factor n off with worst case bad predictions.

Preferential Round Robin

- **Preferential Round Robin**
  - Run RR and SPJF in parallel!
  - SPJF at a rate $1-\lambda$.
  - RR at a rate λ
- Competitive ratio: $\min\{\frac{1}{1-\lambda}\left(1+\frac{2\eta}{n}\right), \frac{2}{\lambda}\}$.
- $\frac{2}{\lambda}$-robust.
- $\frac{2-\lambda}{2-2\lambda}$-consistent. (requires detailed analysis)

Analysis

Definition

An algorithm is *monotonic* if given two instances handling processing times $(x_1, \ldots, x_n)$ and $(x'_1, \ldots, x'_n)$ s.t. $x_i \leq x'_i \ \forall i$, then the objective function f satisfies $f(x_1, \ldots, x_n) \leq f(x'_1, \ldots, x'_n)$.

Lemma 1

Given two monotonic algorithms A and B with competitiveness α and β (respectively) for the minimum makespan problem with preemptions, and a parameter $\lambda \in (0, 1)$, one can obtain a competitive ratio of $\min\{\frac{\alpha}{\lambda}, \frac{\beta}{1-\lambda}\}$

Analysis

Proof of Lemma 1.

By running A and B in parallel, each unit of time is split among the two in a $(\lambda, 1-\lambda)$ split. This would increase the makespan of each one alone by the respective factors: $1/\lambda$ and $1/(1-\lambda)$.

The execution of any algorithm only decreases the remaining processing time of jobs. By monotonicity, this may only reduce the makespan of the other algorithm.

Thus both bounds hold simultaneously. □

Analysis - Competitiveness of SPJF

Lemma 2 Competitiveness of SPJF

The SPJF algorithm has competitive ratio at most $(1 + \frac{2\eta}{n})$

Proof. Assume w.l.o.g. $x_1 \leq \ldots \leq x_n$ For any (i, j) let $d(i, j)$ be the amount of job i executed before completion of job j.

- If $i < j$ and $y_i < y_j \Rightarrow d(i, j) + d(j, i) = x_i + 0$,
- If $i < j$ and $y_i \geq y_j \Rightarrow d(i, j) + d(j, i) = 0 + x_j$

$$ALG = \sum_{j=1}^{n} x_j + \sum_{(i,j):i<j} (d(i, j) + d(j, i))$$

$$= \sum_{j=1}^{n} x_j + \sum_{\substack{(i,j):i<j \\ y_i<y_j}} x_i + \sum_{\substack{(i,j):i<j \\ y_i \geq y_j}} x_j = \sum_{j=1}^{n} x_j + \sum_{(i,j):i<j} x_i + \sum_{\substack{(i,j):i<j \\ y_i \geq y_j}} (x_j - x_i)$$

Analysis - Competitiveness of SPJF

$$
\begin{aligned}
ALG &\leq \sum_{j=1}^{n} x_j + \sum_{(i,j):i<j} x_i + \sum_{\substack{(i,j):i<j \\ y_i \geq y_j}} (\eta_i + \eta_j) \\
&= OPT + \sum_{\substack{(i,j):i<j \\ y_i \geq y_j}} (\eta_i + \eta_j) \\
&\leq OPT + (n-1)\eta
\end{aligned}
$$

$$
x_j \geq 1 \, \forall j \Rightarrow OPT \geq \frac{n(n+1)}{2}
$$

$$
\Rightarrow \frac{ALG}{OPT} \leq 1 + \frac{2(n-1)\eta}{n(n+1)} < 1 + \frac{2\eta}{n} \qquad \square
$$

Analysis

Theorem: Competitiveness of preferential RR.

The preferential RR algorithm with $\lambda \in (0,1)$ has competitive ratio at most $\min\{\frac{1}{1-\lambda}\left(1+\frac{2\eta}{n}\right), \frac{2}{\lambda}\}$. In particular, it is $\frac{2}{\lambda}$-robust and $\frac{1}{1-\lambda}$-consistent.

Proof.

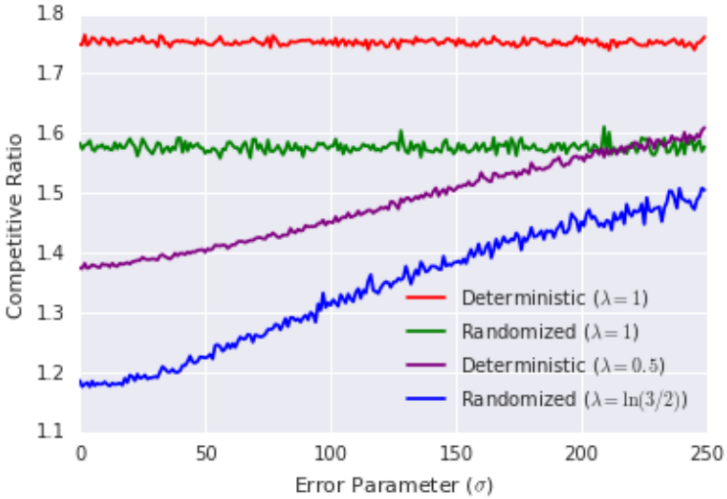
- Recall for SPJF, $c = 1 + \frac{2\eta}{n}$ and for RR $c = 2$.
- Combining both algorithms and apply the result from Lemma 1. □.

Theorem

The competitive ratio of preferential RR under perfect prediction is at most $\frac{2-\lambda}{2-2\lambda}$

Proof ommited.

Skii rental – Experimental results



Non-clairvoyant scheduling –Experimental results

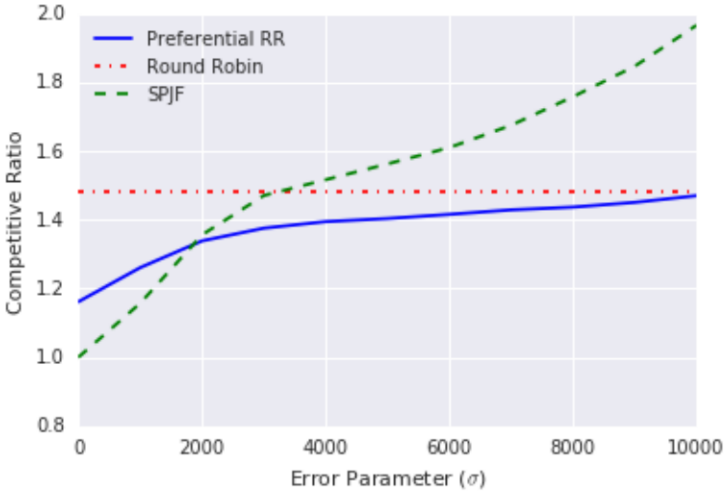


Figure: Non-clairvoyant Scheduling

Conclusions

Conclusions

Conclusions

- Interesting examples of extending online algorithms when predictions are available.
- No systematic approach but a key idea: use a hyperparameter to represent the level of trust in the prediction accuracy.
- How to use the predictions is case-dependent.
- At least in the Skii rental case, randomization helps.

Thank You

Thank You for listening!

Questions?