Stanford MS&E 215 / CS 264:
Algorithms with predictions

Ellen Vitercik*
vitercik@stanford.edu

May 27, 2025

1 Motivation

These lecture notes describe a line of research on *algorithms with predictions* [3] that attempt to achieve nearly optimal performance on "natural" inputs without sacrificing worst-case guarantees. This line of research is built on the key observation that for natural inputs, we may have some predictions about the optimal solutions (for example, that they resemble instances that we have encountered in the past). These lecture notes overview a few easy-to-follow approaches, but many papers have been written on this topic in the past few years, which have been cataloged on the website `algorithms-with-predictions.github.io`.

2 Online algorithms

Much of the literature on algorithms with predictions has focused on *online* problems, where the algorithm's full input is not revealed upfront but only gradually or at some later stage. For example, in caching, memory requests arrive over time, and we must decide what to keep in the cache. However, we may have some predictions about the next time an element will be requested, which we can use to fine-tune our caching policy [2]. Similarly, job lengths may be unknown when scheduling jobs on a machine until the job terminates, but nonetheless, we must decide which jobs to schedule when. However, we may reasonably have some predictions about the job lengths before they begin [5].

2.1 Rent-or-buy (the ski rental problem)

The ski rental problem has emerged as a common test case in the algorithms-with-predictions literature, used to test new ideas and methodologies. It models an abstract, common problem involving a decision: incur a recurring expense or pay a one-time fee that eliminates the ongoing cost. The problem is phrased in terms of a skier who will ski for an unknown number of days (they do not know if they will like skiing or how long the season will be). They can rent skis each day for \$1 a day or buy for \$$b$. For example, if they rent for five days and then buy, they will pay $5 + b$ dollars. If the person was clairvoyant and knew they

[*] These notes are course material and have not undergone formal peer review. Please feel free to send me any typos or comments.

would ski for exactly x days, they should rent if $x < b$ and buy immediately on the first day otherwise. Thus, the optimal clairvoyant strategy pays $\mathrm{OPT} = \min\{x, b\}$.

Under the *breakeven algorithm*, the skier rents for $b - 1$ days, and if they want to keep skiing, buy on the b^{th} day. Under this algorithm, the skier will pay x if $x < b$ and $b - 1 + b = 2b - 1$ if $x \geq b$, so $\mathrm{ALG} = x\mathbf{1}_{\{x<b\}} + (b - 1 + b)\mathbf{1}_{\{x\geq b\}}$. To evaluate an online algorithm, we typically bound its *competitive ratio (CR)*, which is the ratio between ALG and OPT. For the breakeven algorithm,

$$\mathrm{CR} = \frac{\mathrm{ALG}}{\mathrm{OPT}} = \frac{x\mathbf{1}_{\{x<b\}} + (b - 1 + b)\mathbf{1}_{\{x\geq b\}}}{\min\{x, b\}}.$$

By a simple case analysis, it is not hard to show that $\mathrm{CR} \leq 2$. Moreover, there is no deterministic algorithm with a worst-case competitive ratio better than 2 [1].

2.2 Ski rental with predictions

We now describe a formulation of ski rental with predictions proposed by Purohit et al. [5]. Suppose the skier has a prediction y of the number of days he will ski with an error bound of $|x - y| \leq \eta$. In the algorithms-with-predictions literature, the goal is to design an algorithm with the following guarantees [2]:

α-consistency: As $\eta \to 0$, the algorithm's competitive ratio converges to some value α. Ideally, α should be close to 1, meaning the algorithm becomes nearly optimal as the predictions improve.

β-robustness: No matter how large η is, the algorithm's competitive ratio is bounded by some β. The value of β should be independent of η, and ideally should not be too much larger than the best-known worst-case competitive ratio (which is 2 for the ski rental problem).

We will warm up with a naive algorithm, which buys immediately if $y > b$ and otherwise rents for however long the skier ends up skiing. In other words, this naive algorithm blindly follows the prediction. The issue with this algorithm is best illustrated by the case where η is enormous, $x > b$, and $y = 1$. In this case,

$$\mathrm{CR} = \frac{\mathrm{ALG}}{\mathrm{OPT}} = \frac{x}{b} \leq \frac{y + \eta}{b} = \frac{1 + \eta}{b}.$$

In other words, we cannot satisfy β-robustness for any β independent of η, except $\beta = \infty$.

Algorithm 1 Ski rental with predictions algorithm by Purohit et al. [5]

```
Input: Prediction $y$, parameter $\lambda \in [0, 1]$.
if $y \geq b$ then
    Buy on the start of day $\lceil \lambda b \rceil$
else
    Buy on the start of day $\lceil \frac{b}{\lambda} \rceil$.
```

This warm-up motivates the algorithm by Purohit et al. [5], Algorithm 1, which is defined by a parameter $\lambda \in [0, 1]$. To interpret this algorithm, it is helpful to analyze the extreme cases where $\lambda \in \{0, 1\}$:

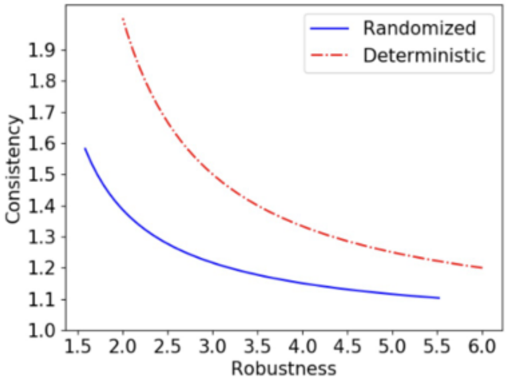


Figure 1: Figure by Purohit et al. [5] illustrating the tradeoff between consistency and robustness for the ski rental problem. Check out the paper for details about the randomized algorithm, which is not covered in these lecture notes.

- If $\lambda = 1$, Algorithm 1 can be simplified as follows: buy on the start of day b (if the skier is still skiing). This is the breakeven algorithm with a competitive ratio of 2.
- If $\lambda = 0$, Algorithm 1 buys immediately if $y \geq b$, and otherwise rents for however long the skier ends up skiing. This corresponds to blindly following the predictions.

Thus, λ can be viewed as a parameter the user can tune based on how much they "trust" the prediction, with $\lambda = 1$ corresponding to zero trust and $\lambda = 0$ corresponding to complete trust.

Purohit et al. [5] prove the following guarantee.

Theorem 2.1. *The competitive ratio of Algorithm 1 is bounded as follows:*

$$CR \leq \min\left\{\frac{1+\lambda}{\lambda}, 1+\lambda+\frac{\eta}{(1-\lambda)OPT}\right\}.$$

We can see that Algorithm 1 is $(1+\lambda)$-consistent because as $\eta \to 0$, this bound approaches $1+\lambda$. Meanwhile, Algorithm 1 is $\frac{1+\lambda}{\lambda}$-robust because no matter how large η is, the competitive ratio is bounded by $\frac{1+\lambda}{\lambda}$. By varying λ, we obtain a tradeoff between consistency and robustness, illustrated by Figure 1.

Proof sketch of Theorem 2.1. This proof follows a case analysis. In the first case, $y \geq b$, in which case Algorithm 1 buys on the start of day $\lceil \lambda b \rceil$. If $x < \lceil \lambda b \rceil$, then both Algorithm 1 and the optimal clairvoyant strategy will rent all days, leading to a competitive ratio of

$$\text{CR} = \frac{\text{ALG}}{\text{OPT}} = \frac{x}{x} = 1. \tag{1}$$

Next, if $\lceil \lambda b \rceil \leq x \leq b$, Algorithm 1 will rent for $\lceil \lambda b \rceil - 1$ days and then buy, paying a total of $\lceil \lambda b \rceil - 1 + b$. However, the optimal clairvoyant strategy will rent all days, leading to a competitive ratio of

$$\text{CR} = \frac{\text{ALG}}{\text{OPT}} = \frac{\lceil \lambda b \rceil - 1 + b}{x}. \tag{2}$$

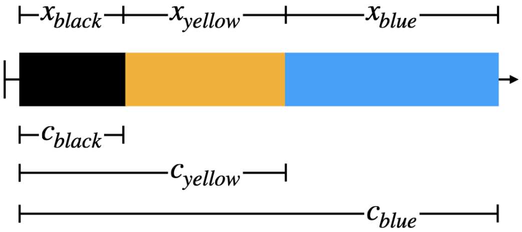


Figure 2: Three jobs with processing times x_{black}, x_{yellow}, and x_{blue}. When the black job is scheduled first, the yellow job is scheduled second, and the blue job is scheduled third, their completion times are c_{black}, c_{yellow}, and c_{blue}.

Finally, if $x \geq b$, then the optimal clairvoyant strategy will buy on the start of day b, leading to a competitive ratio of

$$\mathrm{CR} = \frac{\mathrm{ALG}}{\mathrm{OPT}} = \frac{\lceil \lambda b \rceil - 1 + b}{b}. \tag{3}$$

Thus, when $y \geq b$ we have that

$$\mathrm{CR} = \frac{\mathrm{ALG}}{\mathrm{OPT}} = \begin{cases} \frac{x}{x} & \text{if } x < \lceil \lambda b \rceil \\ \frac{\lceil \lambda b \rceil - 1 + b}{x} & \text{if } \lceil \lambda b \rceil \leq x \leq b \\ \frac{\lceil \lambda b \rceil - 1 + b}{b} & \text{if } x \geq b. \end{cases}$$

This is maximized when $x = \lceil \lambda b \rceil$, in which case

$$\mathrm{CR} = \frac{b + \lceil \lambda b \rceil - 1}{\lceil \lambda b \rceil} \leq \frac{1 + \lambda}{\lambda}.$$

The case where $y < b$ follows by a similar case analysis. □

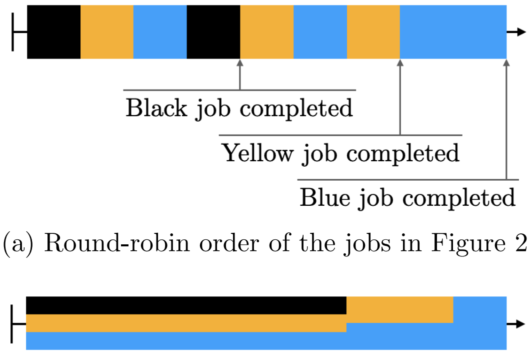
3 Job scheduling

Next, we move on to a simple single-machine job scheduling problem. The task is to schedule n jobs on a single machine. Each job has an unknown processing time x_j. The goal is to minimize the sum of the jobs' completion times. In other words, if job j completes at time c_j, the goal is to minimize $\sum c_j$. The job completion times are illustrated in Figure 2. In this problem, any job can be preempted at any time and resumed at a later time at no cost.

If the processing times are known, the optimal solution is simple: schedule the jobs in increasing order of x_j. If, without loss of generality, the jobs are ordered as $x_1 \leq \cdots \leq x_n$, then this optimal solution has a total completion time of

$$\mathrm{OPT} = \sum_{i=1}^{n} \sum_{j=1}^{i} x_i.$$

Meanwhile, if the processing times are unknown, a simple round-robin algorithm achieves a competitive ratio of 2 [4]. This algorithm schedules one unit of time per remaining job, round-robin, until all jobs have been completed, as illustrated in Figure 3a. In what follows, it will be useful to think of the round-robin algorithm in its equivalent, continuous form: if we are scheduling k jobs round-robin, we can think of this as being equivalent to running all k jobs simultaneously at a rate of $\frac{1}{k}$. This is illustrated in Figure 3b.



(a) Round-robin order of the jobs in Figure 2.

(b) Continuous version of the round-robin order. Instead of scheduling the black job, for example, round-robin for 2 discrete timesteps, we schedule it in parallel with the two other jobs for six timesteps at a rate of $\frac{1}{3}$.

Figure 3: Illustration of round-robin scheduling.

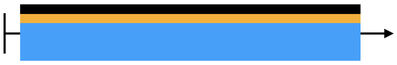


Figure 4: Illustration of preferential round-robin under the following conditions: $\lambda = \frac{1}{2}$, there are three jobs, and the blue job has the shortest predicted processing time. The black and the yellow jobs are run at a rate of $\frac{1-\lambda}{3} = \frac{1}{6}$ and the blue job is run at a rate of $\lambda + \frac{1-\lambda}{3} = \frac{2}{3}$.

Scheduling with predictions

Suppose we have predictions $y_1, \dots, y_n$ of $x_1, \dots, x_n$ with $\sum |y_i - x_i| \leq \eta$. As in Section 2.2, Purohit et al. [5] define an algorithm that interpolates between two extremes: (1) the worst-case round-robin (RR) algorithm, and (2) blindly following the predictions. If we completely trust the predictions, we should schedule jobs in increasing order of y_i. We refer to this algorithm as *shortest prediction job first (SPJF).*

To interpolate between these extremes, Purohit et al. [5] propose an algorithm called *preferential round-robin*, which is defined by a parameter $\lambda \in [0, 1]$. This algorithm runs SPJF and RR simultaneously, SPJF at a rate of λ and RR at a rate of $1 - \lambda$. To illustrate, suppose there are k jobs remaining, and the job with the shortest predicting processing time is job i. Preferential round-robin will run each of the remaining jobs other than job i simultaneously at a rate of $\frac{1-\lambda}{k}$. Meanwhile, it will run job i at a rate of $\frac{1-\lambda}{k} + \lambda$. This is illustrated by Figure 4.

Purohit et al. [5] show that preferential round-robin's competitive ratio is bounded as follows:

$$\text{CR} \leq \min\left\{\frac{1}{\lambda}\left(1 + \frac{2\eta}{n}\right), \frac{1}{1-\lambda} \cdot 2\right\}. \tag{4}$$

Thus, this algorithm is $\frac{1}{\lambda}$-consistent and $\frac{2}{1-\lambda}$-robust. To help interpret Equation (4), we observe that $1+\frac{2\eta}{n}$ is the CR of SPJF, whereas 2 is the CR of RR, so Equation (4) interpolates between these two CRs.

References

[1] Anna R Karlin, Mark S Manasse, Larry Rudolph, and Daniel D Sleator. Competitive snoopy caching. *Algorithmica*, 3:79–119, 1988.

[2] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *International Conference on Machine Learning (ICML)*, 2018.

[3] Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In Tim Roughgarden, editor, *Beyond the worst-case analysis of algorithms*. Cambridge University Press, 2020.

[4] Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47, 1994.

[5] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Conference on Neural Information Processing Systems (NeurIPS)*, pages 9661–9670, 2018.