Algorithms and Uncertainty, Winter 2018/19 Lecture 1 (4 pages)

Ski Rental

Instructor: Thomas Kesselheim

We will first consider the *ski-rental problem.* Suppose you stay in a ski resort for T days. You can either rent skis (cost 1 per day) or buy them (cost B once). Unfortunately, you do not which option is cheaper because you do not know the number of days you will go skiing in advance. Every morning, you get to know if you will go skiing that day. Then, you can choose if you rent skis for that day or if you buy them, in which case you can use them for the rest of the stay and do not have to rent or buy them again.

1 A Deterministic Online Algorithm

There is a very simple online algorithm for this problem: Rent skis up to $B-1$ times. Buy skis on the B-th day that you want to go skiing.

Theorem 1.1. *For any sequence of "skiing"/"not skiing" days, the described online algorithm's cost exceeds the optimal possible cost on this particular sequence by a factor of at most* $2-\frac{1}{B}$.

Proof. Let there be k days of skiing in the sequence. If $k<B$, the optimal possible cost on this sequence is k by always renting. Our algorithms always rents and pays k as well: It makes the optimal choice.

If $k \geq B$, the optimal choice is to buy the skis on the very first day, resulting in cost of B. What is our algorithm's cost? We rent the skis for $B-1$ days and buy them once. So we pay $B-1+B=2B-1$. That is, our cost exceeds the optimal cost by a factor of $\frac{2B-1}{B}=2-\frac{1}{B}$. □

2 Online Competitive Analysis

What have we done so far? We have devised an algorithm that only works online and we compared its performance to what we could have done offline.

In an abstract way, the algorithm operates on an input sequence $\sigma=(\sigma_1,\sigma_2,\dots,\sigma_T)$. In the t-th step, request σ_t arrives and we have to process it only knowing $\sigma_1,\sigma_2,\dots,\sigma_t$ but not $\sigma_{t+1},\sigma_{t+2},\dots,\sigma_T$.

We say that a deterministic algorithm for a minimization problem is α-competitive if

$$c(\mathrm{ALG}(\sigma)) \leq \alpha \cdot c(\mathrm{OPT}(\sigma)) + b \quad \text{for any sequence } \sigma \ , \tag{1}$$

where $c(\mathrm{ALG}(\sigma))$ denotes the cost that the algorithm incurs on sequence σ and $c(\mathrm{OPT}(\sigma))$ denotes the cost of the optimal offline solution. That is, it is the cost that one could have achieved on σ with perfect knowledge and unlimited computational power.

The constant $b \geq 0$ is useful to avoid corner cases. If Equation (1) holds with $b=0$, then the algorithm is *strictly* α-competitive.

Theorem 1.2. *There is no deterministic algorithm for Ski Rental that is strictly* α*-competitive for* $\alpha < 2-\frac{1}{B}$.

Proof. We set $T=2B$. Consider the sequence σ, in which we go skiing every day. Let ℓ denote the number of days that the algorithms rents skis on this sequence, so $0 \leq \ell \leq T$.

We now make a case distinction. The first case is $\ell=T$. In this case, $c(\mathrm{ALG}(\sigma))=\ell=T$ but $c(\mathrm{OPT}(\sigma))=B$. So, we observe that for this particular sequence

$$\frac{c(\mathrm{ALG}(\sigma))}{c(\mathrm{OPT}(\sigma))}=2$$

and therefore Condition (1) only holds for $\alpha \geq 2$.

The second case is $\ell < T$. That is, on sequence σ, the algorithm rents the skis for the first ℓ days and buys them on day $\ell+1$. Its cost is $c(\mathrm{ALG}(\sigma)) = \ell+B$. The algorithm works online, that is, it makes the exact same decisions regardless of the values $\sigma_{\ell+2}, \dots, \sigma_T$. Let σ' be the sequence in which we go skiing for the first $\ell+1$ days and never afterwards. As $\sigma_1 = \sigma'_1, \dots, \sigma_{\ell+1} = \sigma'_{\ell+1}$, the algorithm does exactly the same in these steps. So $c(\mathrm{ALG}(\sigma')) = \ell + B$. However, $c(\mathrm{OPT}(\sigma')) = \min\{\ell+1, B\}$.

For this sequence σ', we have

$$\frac{c(\mathrm{ALG}(\sigma'))}{c(\mathrm{OPT}(\sigma'))} = \frac{\ell+B}{\min\{\ell+1, B\}} \geq 2 - \frac{1}{B} \ ,$$

where the last step follows from a simple case distinction.

Overall, we have shown that for any algorithm, there is a sequence σ such that

$$c(\mathrm{ALG}(\sigma)) \geq \left(2 - \frac{1}{B}\right) \cdot c(\mathrm{OPT}(\sigma)) \ ,$$

which means that Condition (1) cannot hold for any $\alpha < 2 - \frac{1}{B}$. □

It is crucial for this perspective that the algorithm is deterministic. We will now turn to randomized algorithms.

3 Randomized Online Algorithms

How can we do better? While deterministic algorithms cannot beat $2-\frac{1}{B}$, randomized algorithms can. First, we have to clarify what this means. If an algorithm is randomized, it may flip coins to guide its decisions. *We assume that the sequence does not adapt to these coin flips.* The cost of an algorithm on a sequence now becomes a random variable. For our benchmark, we use its expectation.

We say that a randomized algorithm for a maximization problem is α-competitive if

$$\mathbf{E}\left[c(\mathrm{ALG}(\sigma))\right] \leq \alpha \cdot c(\mathrm{OPT}(\sigma)) + b \quad \text{for any sequence } \sigma \ ,$$

where σ may not depend on the internal randomness of the algorithm.

To understand the idea, let us consider the following very simple randomized algorithm. We flip a coin: With probability $\frac{1}{2}$ we proceed as before and buy the skis on day B; otherwise (so also with probability $\frac{1}{2}$), we buy the skis already on day $\frac{3}{4}B$.

Theorem 1.3. *The randomized algorithm for ski rental is strictly $\frac{15}{8}$-competitive.*

Proof. We proceed as before. Again, consider an arbitrary sequence σ. We will show that $\mathbf{E}\left[c(\mathrm{ALG}(\sigma))\right] \leq \frac{15}{8} c(\mathrm{OPT}(\sigma))$. To this end, we distinguish three cases regarding the number of skiing days k in the sequence.

If $k < \frac{3}{4}B$, we always rent skis, so $\mathbf{E}\left[\mathrm{ALG}(\sigma)\right] = \mathrm{OPT}(\sigma) = k$.

If $k \geq B$, then $\mathrm{OPT}(\sigma) = B$. Our algorithm eventually always buys skis. It rents skis either $B-1$ times or only $\frac{3}{4}B - 1$ times, depending on the outcome of the random coin flip. So, $\mathbf{E}\left[c(\mathrm{ALG}(\sigma))\right] = \frac{1}{2}(B - 1 + B) + \frac{1}{2}(\frac{3}{4}B - 1 + B) \leq \frac{15}{8}B = \frac{15}{8}c(\mathrm{OPT}(\sigma))$.

If $\frac{3}{4}B \leq k < B$, we have $\mathrm{OPT}(\sigma) = k$. With probability $\frac{1}{2}$, the algorithm rents skis all k times. With probability $\frac{1}{2}$, it rents the skis $\frac{3}{4}B - 1$ times and buys them in the following step. So $\mathbf{E}\left[c(\mathrm{ALG}(\sigma))\right] = \frac{1}{2}k + \frac{1}{2}(\frac{3}{4}B - 1 + B) \leq \frac{1}{2}k + \frac{1}{2}(k + \frac{4}{3}k) = \frac{5}{3}k \leq \frac{15}{8}k = \frac{15}{8}c(\mathrm{OPT}(\sigma))$. □

It is important to remark at this point that this argument only works because the sequence does not depend on the coin flip; this is where the improvement comes from. Furthermore, this is clearly not the best randomized algorithm in terms of the competitive ratio. We will see a much more structured and general approach soon.

4 Other Models of Uncertainty

Competitive analysis is very robust: We essentially do not make any assumptions regarding the input sequence. Consequently, our assumptions cannot be wrong. Whatever underlying process actually generates the sequence, it can always be captured. Every *positive* guarantee that we show will hold. However, it is also very pessimistic. It assumes that we do not have any prior knowledge on the input.

In the ski-rental problem, we motivated the uncertainty by the fact that we do not know in advance how much we like skiing or what the weather will be like. In either case, we probably do have some information in advance. After going skiing once or twice, we learn how much we like it and how often we will probably go again. A weather forecast may not be entirely accurate but also gives us information to be taken into consideration.

Later in the course, we will see several different examples of such models. Let us assume that every day in the morning somebody flips a coin. If it shows heads, we want to go skiing, otherwise we don't. To be more precise, we assume that $\sigma_1, \dots, \sigma_T$ are independent random variables. For every step t we have $\sigma_t = 1$ with probability q and $\sigma_t = 0$ otherwise. We know q in advance. *Importantly, now the sequence itself is random.*

We will now derive the *optimal online algorithm*. Note that this not $\mathrm{OPT}(\sigma)$ from above, which is the optimal offline solution. It is defined as follows. For every fixed q, some algorithm ALG^* minimizes the expected cost $\mathbf{E}\left[c(\mathrm{ALG}^*(\sigma))\right]$.

Let $C(T) = \min_{\mathrm{ALG}} \mathbf{E}\left[c(\mathrm{ALG}(\sigma_1, \dots, \sigma_T))\right]$ denote the smallest expected cost on a sequence of length T of any online algorithm. We will now derive a recursive expression for $C(T)$. Clearly, $C(0) = 0$. Now, consider the first step of our algorithm. If $\sigma_1 = 1$, we have to decide to buy the skis or to rent them. If $\sigma_1 = 0$, there is no decision yet. The key observation is that if ALG^* does not buy skis now, it will face the exact same decision process with $T - 1$ steps. For this reason

$$C(T) = \begin{cases} q(C(T-1)+1) + (1-q)C(T-1) & \text{if } \mathrm{ALG}^* \text{ rents on the first day when } \sigma_1 = 1 \\ qB + (1-q)C(T-1) & \text{otherwise} \end{cases}$$

The algorithm ALG^* is defined to choose the cheaper of the two options. So

$$C(T) = q \min\{C(T-1)+1, B\} + (1-q)C(T-1) \ .$$

Let us understand this recursion. We start from $C(0) = 0$. Initially, we will have $C(T-1)+1 \leq B$ and so $C(T) = C(T-1) + q$, which means $C(T) = qT$ for the first values of T. At some point τ we have $C(T-1) + 1 > B$ for the first time. More precisely, this is the smallest τ such that $q(\tau - 1) + 1 > B$, which is $\tau = \left\lfloor \frac{B-1}{q} + 2 \right\rfloor$.

Note that $C(T-1) + 1 > B$ to be true for larger T. That is, for all $T \geq \tau$, we have $C(T) = qB + (1-q)C(T-1)$.

One can solve this recursion further from here (see Figure 1 for the solution). However, this is not necessary to derive the optimal algorithm. Recall that the minimum in the recursion comes from the choice of the algorithm on the first day. We now know that, in the first step, if $\sigma_1 = 1$, the optimal algorithm will buy the skis if $T \geq \tau$ and rent them otherwise. If $\sigma_1 = 0$ it recursively runs the algorithm for $T - 1$. In other words, the optimal algorithm waits for the first t with $\sigma_t = 1$. Then, if $T - t + 1 \geq \tau$, it buys the skis. Otherwise, it rents them throughout the sequence.

So, interestingly, this algorithm does quite the opposite of what was good in competitive analysis: It never buys the skis after renting them once. The algorithm that does well in competitive analysis always rents the skis a number of times before buying them.

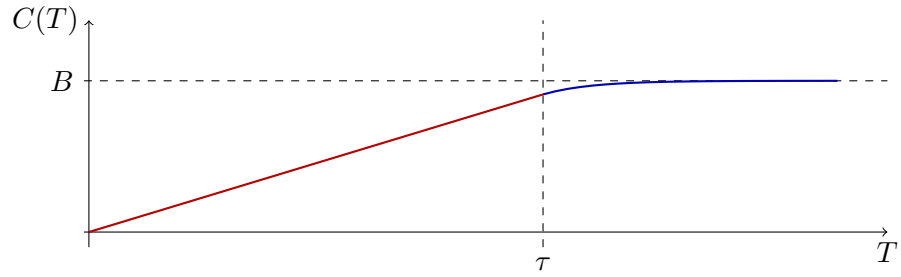

Figure 1: Values of $C(T)$ for $B = 10$, $q = \frac{1}{10}$. Right of τ, the function asymptotically approaches B (but never actually reaches it).

5 Outlook

We have already seen two models of uncertainty today. On the one hand, there is competitive analysis, in which one does not know anything about the input in advance. On the other hand, our stochastic model is an example of a Markov decision process. Here, one knows exactly the stochastic process that generates the input. We will see more examples of these models as well as other ones throughout the course.