Secretary Problems

Petropanagiotaki Maria

January 15 2015

MPLA, Algorithms & Complexity 2

Simplest form of the problem

1. The candidates are totally ordered from best to worst with no ties.
2. The candidates arrive sequentially in random order.
3. We can only determine the relative ranks of the candidates interviewed so far. We cannot observe the absolute ranks.
4. Our goal is to choose the very best candidate.
5. Once a candidate is rejected cannot be recalled.
6. The number of candidates n is known to us.

Algorithm

Let a certain number $k-1$ of the candidates go by, and then select the first candidate you see who is better than all of the previous candidates (if she exists).
If she does not exist, accept the last candidate.

But which is the right k?
In order to find it, we need to calculate the probability of success and then the optimal k is one that maximizes this probability.

Algorithm

Let a certain number $k-1$ of the candidates go by, and then select the first candidate you see who is better than all of the previous candidates (if she exists).
If she does not exist, accept the last candidate.

But which is the right k?
In order to find it, we need to calculate the probability of success and then the optimal k is one that maximizes this probability.

Probability

Let X_n denote the number of the best candidate and $S_{n,k}$ denote the event of success.

$$P(S_{n,k} \mid X_n = i) = \begin{cases} 0, & i \in \{1, 2, ..., k-1\}; \\ \frac{k-1}{i-1} & i \in \{k, k+1, ..., n\}. \end{cases}$$

The probability of success $p_n(k)$ is:

- If $k = 1$ then $p_n(k) = P(S_{n,k}) = 1/n$
- If $k > 1$ then

$$p_n(k) = P(S_{n,k}) = \sum_{i=1}^{n} P(X_n = i) P(S_{n,k} \mid X_n = i) = \sum_{i=k}^{n} \frac{1}{n}\frac{k-1}{i-1} = \frac{k-1}{n}\sum_{i=k}^{n}\frac{1}{i-1}$$

Probability

Let X_n denote the number of the best candidate and $S_{n,k}$ denote the event of success.

$$P(S_{n,k} \mid X_n = i) = \begin{cases} 0, & i \in \{1, 2, ..., k-1\}\,; \\ \frac{k-1}{i-1} & i \in \{k, k+1, ..., n\}\,. \end{cases}$$

The probability of success $p_n\left(k\right)$ is:

- If $k = 1$ then $p_n\left(k\right) = P\left(S_{n,k}\right) = 1/n$
- If $k > 1$ then
  $p_n\left(k\right) = P\left(S_{n,k}\right) = \sum_{i=1}^{n} P\left(X_n = i\right) P(S_{n,k} \mid X_n = i) = \sum_{i=k}^{n} \frac{1}{n}\frac{k-1}{i-1} = \frac{k-1}{n}\sum_{i=k}^{n}\frac{1}{i-1}$

Probability

Let X_n denote the number of the best candidate and $S_{n,k}$ denote the event of success.

$$P(S_{n,k} \mid X_n = i) = \begin{cases} 0, & i \in \{1, 2, ..., k-1\}\,; \\ \frac{k-1}{i-1} & i \in \{k, k+1, ..., n\}\,. \end{cases}$$

The probability of success $p_n(k)$ is:

- If $k = 1$ then $p_n(k) = P\left(S_{n,k}\right) = 1/n$
- If $k > 1$ then $p_n(k) = P\left(S_{n,k}\right) = \sum_{i=1}^{n} P\left(X_n = i\right) P(S_{n,k} \mid X_n = i) \quad = \sum_{i=k}^{n} \frac{1}{n}\frac{k-1}{i-1} \quad = \frac{k-1}{n}\sum_{i=k}^{n}\frac{1}{i-1}$

Optimal k

$$p_n(k) = \frac{k-1}{n} \sum_{i=k}^{n} \left(\frac{n}{i-1} \right) \left(\frac{1}{n} \right)$$

We recognize the sum above as the left *Riemann sum* for the function $f(t) = \frac{1}{t}$ corresponding to the partition of the interval $\left[\frac{k-1}{n}, 1\right]$ into $(n-k)+1$ sub-intervals of length $\frac{1}{n}$ each.
It follows that

$$p_n(k) \approx -\frac{k-1}{n} \ln \frac{k-1}{n}$$

If $\frac{k}{n} \to x \in (0,1)$ as $n \to \infty$ then the expression on the right converges to

$$-x \ln x \text{ as } n \to \infty$$

So, the optimal x is $1/e$ and the algorithm is e-competitive.

What is c-competitive?

- An *online algorithm* is an algorithm which must satisfy an unpredictable sequence of requests, completing each request without being able to see the future.
- An *offline algorithm* is an algorithm that can view the sequence of requests in advance.
- *Competitive ratio* of an online algorithm is defined as the maximum, over all possible input sequences, of the ratio between the cost incurred by the online algorithm and the cost incurred by an optimal offline algorithm. An optimal online algorithm is then, one whose competitive ratio is least.

What is c-competitive?

- An *online algorithm* is an algorithm which must satisfy an unpredictable sequence of requests, completing each request without being able to see the future.
- An *offline algorithm* is an algorithm that can view the sequence of requests in advance.
- *Competitive ratio* of an online algorithm is defined as the maximum, over all possible input sequences, of the ratio between the cost incurred by the online algorithm and the cost incurred by an optimal offline algorithm. An optimal online algorithm is then, one whose competitive ratio is least.

Multiple-choice secretary problem

- We want, now, to choose k secretaries.
- There are two algorithms, the *Virtual* and the *Optimistic*.
- Both of them are based on the idea of a sampling period of $t \in \{k+1, ..., n\}$ steps (during which the algorithms passes on all candidates), followed by hiring some of the secretaries for the remaining $n - t$ steps.
- We call t the *thresold time* of the algorithms and we will leave it unspecified for now.
- Let T be the set of sampled elements.
- Let R be a *reference set*, consisting of the k elements with the largest $v(i)$ values seen during the first t steps. These elements are kept for comparison but not selected.
- At any given time, let $j_1, j_2, ..., j_{|R|}$ be the elements of R sorted by decreasing $v(j_i)$.

Multiple-choice secretary problem

- We want, now, to choose k secretaries.
- There are two algorithms, the *Virtual* and the *Optimistic*.
- Both of them are based on the idea of a sampling period of $t \in \{k+1, ..., n\}$ steps (during which the algorithms passes on all candidates), followed by hiring some of the secretaries for the remaining $n - t$ steps.
- We call t the *thresold time* of the algorithms and we will leave it unspecified for now.
- Let T be the set of sampled elements.
- Let R be a *reference set*, consisting of the k elements with the largest $v(i)$ values seen during the first t steps. These elements are kept for comparison but not selected.
- At any given time, let $j_1, j_2, ..., j_{|R|}$ be the elements of R sorted by decreasing $v(j_i)$.

Multiple-choice secretary problem

- We want, now, to choose k secretaries.
- There are two algorithms, the *Virtual* and the *Optimistic*.
- Both of them are based on the idea of a sampling period of $t \in \{k+1, ..., n\}$ steps (during which the algorithms passes on all candidates), followed by hiring some of the secretaries for the remaining $n - t$ steps.
- We call t the *thresold time* of the algorithms and we will leave it unspecified for now.
- Let T be the set of sampled elements.
- Let R be a *reference set*, consisting of the k elements with the largest $v(i)$ values seen during the first t steps. These elements are kept for comparison but not selected.
- At any given time, let $j_1, j_2, ..., j_{|R|}$ be the elements of R sorted by decreasing $v(j_i)$.

Virtual algorithm

- The secretary i is selected if and only if

$$v(i) > v(j_k) \text{ and } j_k < t$$

- Whenever $v(i) > v(j_k)$ element i is added to R, while element j_k is removed from R.

Thus, $|R| = k$ always and i is selected if and only if its value exceeds that of the k^{th} best element seen so far, and the k^{th} best element was seen during the sampling period.

Optimistic algorithm

- The secretary i is selected if and only if

$$v(i) > v(j_{|R|})$$

- Whenever i is selected, $j_{|R|}$ is removed from the set R, but no new elements are ever added to R.

Thus, intuitively, elements are selected when they beat one of the remaining reference points from R.

Multiple-choice secretary problem

We first observe that neither algorithm ever selects more than k secretaries.
Each selection involves the removal of a sample $j_i \in R \cap T$ from R, and no elements from T are ever added to R by either algorithm after time t. Since R starts with only k samples, no more than k elements can be selected.

Multiple-choice secretary problem

Lemma

For all $a \leq k$, the probability that the virtual algorithm selects element v_a^* is

$$P\left[i_a^* \in S\right] \geq \frac{t}{n} \ln(n/t)$$

Multiple-choice secretary problem

Proof.

If v_a^* is observed at time $i_a^* = i > t$, it will be selected if and only if the k^{th} smallest element of R at that time was sampled at or before time t. Because the permutation is uniformly random, this happens with probability $t/(i-1)$. Each i is equally likely to be the time at which v_a^* is observed, so the probability of selecting v_a^* is

$$P\left[i_a^* \in S\right] = \sum_{i=t+1}^{n} \frac{1}{n}\frac{t}{i-1} = \frac{t}{n}\sum_{i=t+1}^{n}\frac{1}{i-1} > \frac{t}{n}\int_{t}^{n}\frac{dx}{x} = \frac{t}{n}\ln(\frac{n}{t})$$

$\square$

Multiple-choice secretary problem

Lemma

For all $a \leq k$, the probability that the optimistic algorithm selects element v_a^* is

$$P\left[i_a^* \in S\right] \geq \frac{t}{n}\ln(n/t)$$

Multiple-choice secretary problem

Theorem

The competitive ratio of both the Virtual and the Optimistic Algorithm approaches e as n tends to infinity, when the algorithms sample $t = \left\lfloor \frac{n}{e} \right\rfloor$ elements.

Multiple-choice secretary problem

Proof.

The theorem follows immediately from these two lemmas, as the expected gain of the algorithm is

$$E\left[v(S)\right] \geq \sum_{a=1}^{k} P\left[i_a^* \in S\right] v_a^* > \frac{t}{n} \ln(n/t) v(S^*)$$

$\frac{t}{n} \ln(n/t)$ is maximized for $t = n/e$, and setting $t = \lfloor n/e \rfloor$ gives us that $t/n \to 1/e$ as $n \to \infty$. Thus, the algorithms' competitive ratios approach e as n tends to infinity. $\square$

Knapsack Secretary Problem

- Let $U = \{1, 2, ..., n\}$ be a set of n elements each have non-negative *weight* $w(i)$ and *value* $v(i)$.
  We extend the notations to sets by writting

$$w(S) = \sum_{i \in S} w(i) \text{ and } v(S) = \sum_{i \in S} v(i)$$

- The algorithm will be given a weight-bound W and must select, in an online fashion, a set $S \subseteq U$ of secretaries following knapsack problem

$$\text{Maximize} \sum_{i \in S} v(i) \text{ subject to } \sum_{i \in S} w(i) \leq W$$

- For $i \in U$ we define the *value density* of i to be the ratio

$$\rho(i) = \frac{v(i)}{w(i)}$$

Algorithm

We assume that $W = 1$.
The algorithm begins by sampling a random number $a \in \{0, 1, 2, 3, 4\}$ from the uniform distribution.

If $0 \leq a \leq 3$
then the algorithm sets $k = 3^a$ and runs the k-secretary algorithm (with $t = \lfloor n/e \rfloor$) to select at most k elements.
If the k-secretary algorithm selects an elements i whose weight $w(i)$ is greater than $1/k$, we override this decision and do not select the element.

Algorithm

We assume that $W = 1$.
The algorithm begins by sampling a random number $a \in \{0, 1, 2, 3, 4\}$ from the uniform distribution.

If $0 \leq a \leq 3$
then the algorithm sets $k = 3^a$ and runs the k-secretary algorithm (with $t = \lfloor n/e \rfloor$) to select at most k elements.
If the k-secretary algorithm selects an elements i whose weight $w(i)$ is greater than $1/k$, we override this decision and do not select the element.

Algorithm

If $a = 4$
then the algorithm samples a random $t \in \{1, 2, ..., n\}$ from the binomial distribution $B(n, \frac{1}{2})$.
Let $X = \{1, 2, ..., t\}$ and $Y = \{t + 1, t + 2, ..., n\}$
For every element $i \in X$ the algorithm observes $v(i)$ and $w(i)$ but does not select i.
It then sets $\hat{\rho} = \rho_X^{(1/2)}$ and selects every element $i \in Y$ which satisfies

$$w(i) \leq 3^{-4},\ \rho(i) \geq \hat{\rho} \text{ and } w(S_{<i} \cup \{i\})$$

where $S_{<i}$ denotes the set of elements which where already selected by the algorithm before observing i.