# Principles of Online Decision-Making (CS-303)

## Problem Set 1 - Solutions

## Problem 1

---

(a) We had derived the optimal solution for the secretary problem in class, and seen that (asymptotically for large $n$), the right approach is to observe the maximum score over the first $n/e$ items, and then to stop as soon as this maximum is exceeded.

Suppose now that you know that the score distribution: $x_i \sim \text{unif}(0,1)$, i.i.d. How would you modify the algorithm to maximize the probability of hitting the winner? Will the success probability be higher/lower/the same than for the original version?

---

The goal here was not to devise a provably optimal scheme, but to think qualitatively about the modifications in our assumptions, and how the problem would change, compared to the assumptions we made in class.

First, the criterion for success is the same as before. Therefore, we only ever accept an $x_i$ that is a maximum so far.

Second, the maximum success probability (over all algorithms) must be at least that of the original algorithm, because the original algorithm is a special case of the class of algorithms considered here (i.e., we could run the original algorithm even though we know more).

Third, given that the $x_i$ are i.i.d. *with known law*, past samples are now of no help to predict the future – whereas in the original problem, the past samples were *all we knew* about the probability law governing future $x_i$.

Here, the only thing that matters is whether we have a higher probability of winning by selecting $x_i$ or by continuing the game.

At time $i$, the probability to win by selecting $x_i$ (provided it exceeds the past maximum) is $p_i(x) = \mathbb{P}(\text{no future } x_j (j \in [i+1, n]) \text{ will exceed } x)$. This is $p_i(x) = x^{(n-i)}$, because each of the remaining $x_j$ is below $x$ independently with probability $x$.

So a good heuristic might be the following: at time $i$, if $p_i(x_i) > 1/2$, we stop, otherwise we continue.

This algorithm is not optimal, because the probability of winning by continuing the game is not $1 - p_i(x_i)$. Indeed, the probability of winning by continuing the game is actually smaller, because it is the probability of the intersection event (the maximum will be after $i$) $\cap$ (we will select the maximum among the future $x_j$).

---

(b) Suppose again the same score distribution as above, and it is again known to the player. But assume the player tries to maximize $\mathbb{E}x_Z$ instead of the probability of finding the absolute best. Can you figure out a good policy? You may want to run some simulations to confirm.

---

The most important thing to note here is that when we maximize $\mathbb{E}x_Z$ instead of the probability of scoring the absolute best apartment, it is not the case any more that the optimal rule necessarily waits for a max-so-far. To see this, suppose we are at time $i = n - 1$, and $x_i$ is higher than the median among all samples so far (but is not a new max). It is better to accept this sample (for a 100% chance of beating the median), instead of waiting for $x_n$ (for a fifty-fifty chance of beating the median).

A possible algorithm is to select a threshold $t$ and accept the first $x_i$ which beats this threshold (or $x_n$ if we did not stop before).

Then, the probability of accepting $x_n$ is $t^{n-1}$.
The expected reward given that we stop at $x_n$ is $\frac{1}{2}$.
The expected reward given that we stop before $x_n$ is $\frac{t+1}{2}$.
So our total expected reward is $(1-t^{n-1})\frac{t+1}{2} + t^{n-1}\frac{1}{2} = \frac{1}{2}(1+t-t^n)$.

The expected reward is $\frac{1}{2}$ if $t$ is 0 or 1, which is expected. Furthermore, the reward is concave in $t$, so we can take the derivative to find the optimal $t$. The derivative cancels for $1 - nt^{n-1} = 0$, *i.e.* $t = n^{-\frac{1}{n-1}}$, which means it is the optimal threshold.

One can observe that $\lim_{n\to\infty} n^{-\frac{1}{n-1}} = 1$, this implies that as the time horizon gets longer, the optimal threshold gets closer to 1. (This can be checked by defining $y = n^{-\frac{1}{n-1}}$ and see $\lim_{n\to\infty} \ln y = \lim_{n\to\infty} -\frac{\ln n}{n-1} = 0$ )

The expected reward with this algorithm is in the end $\frac{1}{2}(1+n^{-\frac{1}{n-1}} - n^{-\frac{n}{n-1}})$, which also approaches 1 as $n$ grows.

## Problem 2

---

Prove Lemma 4.4 in L&S.

---

(a) Recall that the cumulative regret $R_n$ is given by $n\mu^* - \mathbb{E}\left[\sum_{t=1}^n X_t\right]$. We can rewrite this as

$$\begin{aligned}
R_n &= n\mu^* - \mathbb{E}\left[\sum_{t=1}^n X_t\right] \\
&= \sum_{t=1}^n (\mu^* - \mathbb{E}[X_t]) \\
&= \sum_{t=1}^n \left(\mu^* - \sum_{i=1}^K \mathbb{E}[X_t | A_t = i]\mathbb{P}(A_t = i)\right) \\
&= \sum_{t=1}^n \sum_{i=1}^K (\mu^* - \mu_i)\mathbb{P}(A_t = i) \\
&= \sum_{t=1}^n \sum_{i=1}^K (\Delta_i)\mathbb{P}(A_t = i)
\end{aligned}$$

where $\Delta_i = \mu^* - \mu_i$. Because both $\Delta_i$ and $\mathbb{P}(A_t = i)$ are non-negative, the entire sum is non-negative. Thus, $R_n \geq 0$.

(b) If $A_t \in \arg\max_a \mu_a$ for all $t$, then $\mathbb{E}[X_t] = \mu^*$ for all $t$. Therefore,

$$R_n = \sum_{t=1}^n (\mu^* - \mathbb{E}[X_t]) = \sum_{t=1}^n (\mu^* - \mu^*) = 0.$$

(c) If $R_n = 0$, then following the derivation of part (a), we must have $\sum_{i=1}^K (\Delta_i)\mathbb{P}(A_t = i) = 0$ for all $t$ (because, for each $t$, the term is nonnegative). Further, for all $i \in [K]$, $\Delta_i \geq 0$ and $\mathbb{P}(A_t = i) \geq 0$. Therefore, to obtain $\sum_{i=1}^K (\Delta_i)\mathbb{P}(A_t = i) = 0$, $\mathbb{P}(A_t = i)$ must be zero for all $i$ such that $\Delta_i > 0$. Therefore, $A_t$ must always be in the set of arms that maximize $\mu_a$, i.e., $A_t \in \arg\max_a \mu_a$ for all $t$. Equivalently, $\mathbb{P}(\mu_{A_t} = \mu^*) = 1$ for all $t$.

## Problem 3

In the given codes, there are two python files: `bandit_algorithm.py` and `bandit_environment.py`. The first file implements the Explore-then-commit algorithm and the $\epsilon$-greedy algorithm. The second file implements the bernoulli bandit environment, and gaussian bandit environment, that you have to complete. We provide `experiment_1.ipynb` to run the algorithms on the environments. Alternatively, you can also run the same code with `experiment_1.py`.

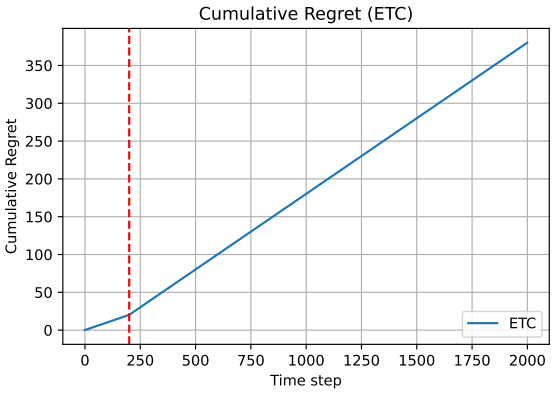
---

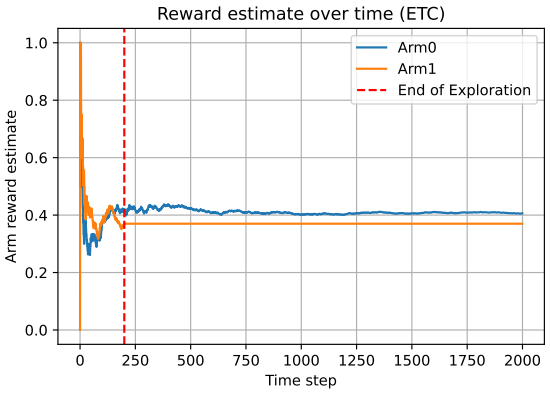
(a) Explore the explore-then-commit algorithm by setting different random seeds.

(b) Try the explore-then-commit algorithm with several seed to find a failing case *i.e.,* the case where the explore-then-commit algorithm chooses the sub-optimal arm at the end of the exploration phase (time step 200). Let $\mu_i$ be the expected reward of arm $i$. Let $\hat{\mu}_i$ be the empirical mean of arm $i$ at the end of the exploration phase. In the code arm 1 is the optimal arm. Under which condition on $\mu_i$ and $\hat{\mu}_i$, does the explore-then-commit algorithm fail?

---

With seed 43170, we can observe the following cumulative regret and estimated rewards of both arms.




The algorithm fails when the empirical mean of the suboptimal arm exceeds that of the optimal arm during the exploration phase, *i.e.*, $\hat{\mu}_0 > \hat{\mu}_1$ is the condition of having this behavior.

---

(c) Let $\hat{p}_{sub}^{200}$ be the empirical probability that the explore-then-commit algorithm chooses the sub-optimal arm at the end of the exploration phase (time step 200). Compute $\hat{p}_{sub}^{200}$ by simulation.

---

With $n\_tries = 10000$, we observed the following empirical probability.

---

(d) Change the sub-optimality gap $\Delta$ and make a plot to see the evolution of $\hat{p}_{sub}^{200}$ as a function of $\Delta$.

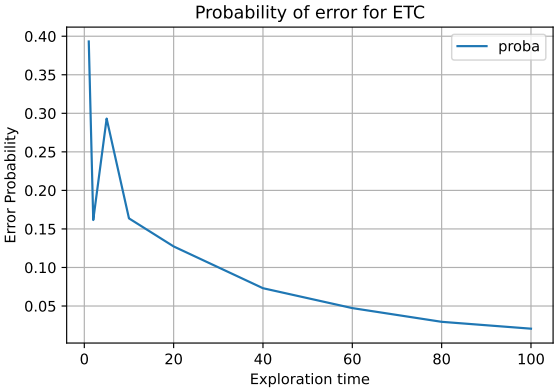
---

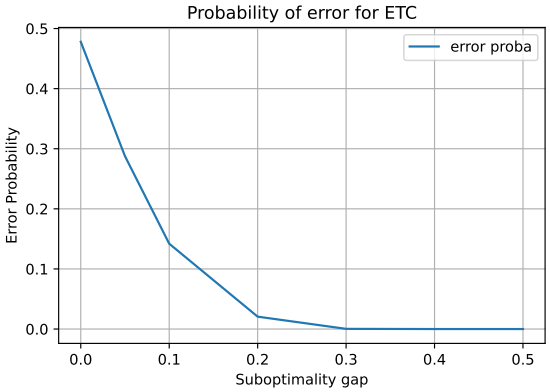
With $n\_tries = 10000$, we observed the following empirical probability.

---

(e) Change the exploration period and make a plot to see the evolution of $\hat{p}_{sub}^{200}$ as a function of the exploration period.

(f) One can consider a variant of the explore-then-commit algorithm that switches arms during the commitment phase if the empirical mean of the arm being pulled becomes lower than that of the other arm. We call this variant the "explore-then-weak-commit" algorithm. Specifically, explore-then-weak-commit algorithm works as follows (with the same notations as in Algorithm 1 on pp. 92 of "Bandit Algorithms" by Lattimore and




Szepesvári):

1. Input $m$ (exploration period per arm)
2. In round $t$ chooses action

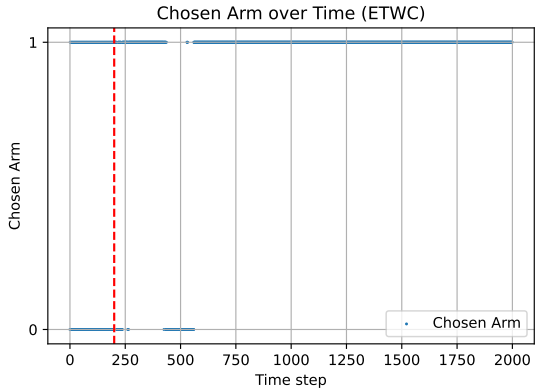
$$A_t = \begin{cases} (t \mod k) & \text{if } t \leq mk; \\ \arg\max_{i \in [k]} \hat{\mu}_i(t-1) & \text{if } t > mk. \end{cases}$$

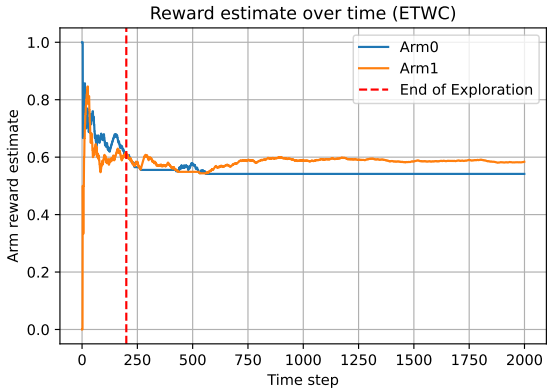
Implement this variant.

(g) Try the explore-then-weak-commit algorithm with seed 14766, explain the behavior you observe with this seed. Give an example of conditions on $\mu_i$ and $\hat{\mu}_i$, under which this kind of behavior can be observed.

---

With seed 14766, we observe:




Here, the algorithm successfully chooses the optimal arm at the end of the exploration phase. However, during the commitment phase, the algorithm switches arms.

This switching-arms behavior during commitment phase can be observed, for example, when

- $\hat{\mu}_0 > \hat{\mu}_1 > \mu_0$;
- $\hat{\mu}_0 > \mu_1$.

In the first case, eventually, the estimated reward of the suboptimal arm will drop below that of the optimal arm during the commitment phase. In the second case, you can observe switching-arms behavior multiple times during the commitment phase. This happens because the estimated reward of both suboptimal and optimal arms are above $\mu_0$, the estimated reward of optimal arm eventually becomes lower than that of the suboptimal arm during the commitment phase, leading to a switch to the suboptimal arm. Then the estimated reward of the suboptimal arm will drop below that of the optimal arm again, leading to a switch back to the optimal arm. This cycle can continue for a while.

---

(h) Can there any runs where the explore-then-weak-commit algorithm performs better than the original explore-then-commit algorithm. If there are some such cases, give the example of conditions of such cases in terms of $\mu_i$ and $\hat{\mu}_i$. (Recall that $\mu_i$ and $\hat{\mu}_i$ are defined in (b).) On contrary, is there any case where original the explore-then-commit algorithm performs better than the explore-then-weak-commit algorithm? If there is such a case, give an example condition on $\mu_i$ and $\hat{\mu}_i$.

---

If $\hat{\mu}_0 > \hat{\mu}_1$, then with standard explore-then-commit algorithm, the algorithm will commit to the suboptimal arm after the exploration period. However, the explore-then-weak-commit algorithm can still switch to the optimal arm during the commitment phase if the estimated reward of the optimal arm becomes higher than that of the suboptimal arm.

On the other hand, if $\hat{\mu}_1 > \hat{\mu}_0 > \mu_0$, then with standard explore-then-commit algorithm, the algorithm will commit to the optimal arm after the exploration period. However, the explore-then-weak-commit algorithm likely to switch to the suboptimal arm during the commitment phase as we observed in (g).

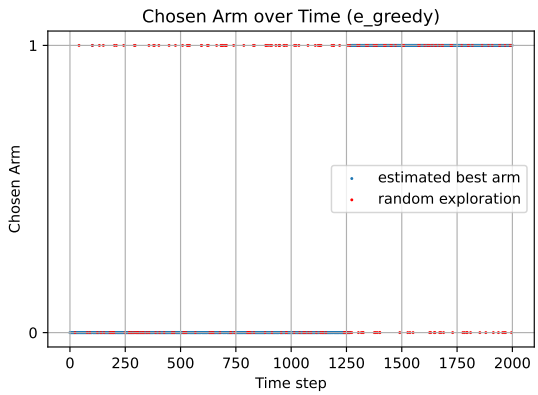
---

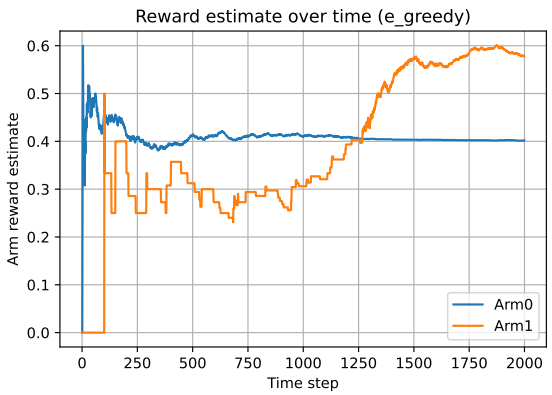
(i) Explore the $\epsilon$-greedy algorithm algorithm by setting different random seeds.

(j) Try the $\epsilon$-greedy algorithm with seed 4195. Are the results similar to those with other seeds? If not, explain what happens here.

---

In (i) we observe typically, within 200 time steps, the algorithm converges to the optimal arm and stays there. However, with seed 4195, we observe much longer time to converge to the optimal arm.



Let $\hat{q}_{sub}^{200}$ be the empirical probability that the $\epsilon$-greedy algorithm estimates the sub-optimal arm (incorrectly) as the optimal arm at time step 200.

(k) Change the sub-optimality gap $\Delta$ and make a plot to see the evolution of $\hat{q}_{sub}^{200}$ as a function of $\Delta$.

With $n\_tries = 10000$, we obtained Fig 1.

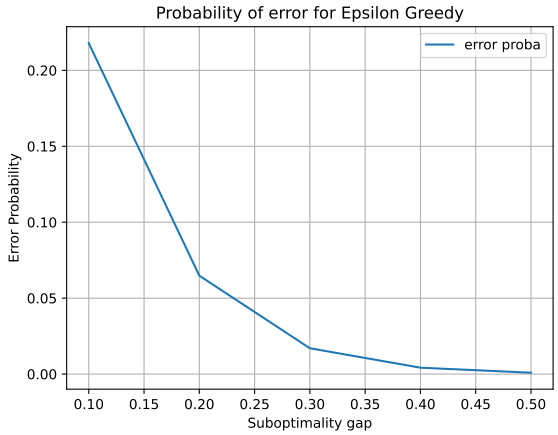



Fig. 1

(l) Change the exploration period and make a plot to see the evolution of $\hat{q}_{sub}^{200}$ as a function of the $\epsilon$.

With $n\_tries = 10000$, we obtained Fig 2.

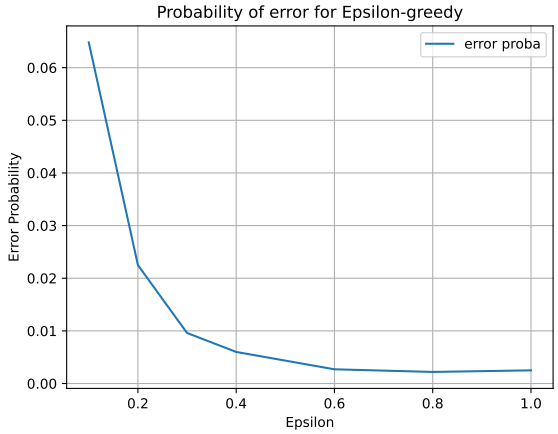



Fig. 2