EE194/290S: ML for Sequential Decision under Uncertainty **Fall 2018**

Lecture 18: Stochastic MAB: Lower bound continued

Lecturer: Vidya Muthukumar *Scribes: Ishan Dikshit, Vasilis Oikonomou*

18.1 Introduction

So far, we have derived upper bounds for the number times we will explore an incorrect arm within the framework of UCB algorithms. Specifically, we found that the number of times we will explore an incorrect arm is $O(\log T)$ where T is the total time the game goes on for. We examined all of the above in the context of Explore then Exploit algorithms and obtained some guarantees on Regret and Pseudo-Regret growth.

Over the course of this lecture we want to know whether it is at all possible to obtain some sort of lower bound for the expected number of times that we should be sampling a sub optimal Arm within some environment. We also want to examine the circumstances under which the need for such a lower bound would arise. We are going to construct a particular problem framework and examine this lower bound for the Explore then Exploit class of algorithms. Subsequently, we are going to generalize this Lower Bound to any general algorithm. Lastly, we are going to examine very basic notions surrounding a different class of Bandit Algorithms: Bayesian Bandits.

18.2 Constructing the Problem Statement

We will explore a simplified version of the Stochastic Multi-armed Bandits problem. Let us assume the following:

1. Rewards from Arm 1 $\sim Bernoulli(\mu_1)$
2. Rewards from Arm 2 $\sim Bernoulli(\mu_2)$
3. μ_1 is a known quantity

Assuming the true value of μ_1 might be a completely unreasonable thing to do in a general 2-Armed Bandit setting. The true value of μ_1 is cosmetic information which we are very unlikely to have access to in practice. Such information is known as **genie information**. Notice that this **does not** reduce the complexity of the problem by a very great extent as we do not know much about μ_2! There is still a fair source of uncertainty in the problem structure. The Genie Information does help us by making us consider a narrower problem structure - it forces us to examine the bottlenecks within the problem.

Now, let us assume (Without Loss of Generality) that in our real environment (reality) $\mu_2 < \mu_1$. In some alternate/fake environment (parallel scenario), however, we might have considered another quantity such that $\mu_2' > \mu_1$. We do not know which reality we exist in. For us, $\mu_2 < \mu_1$ and $\mu_2 > \mu_1$ are both real possibilities. We are considering the case that our reality is one such that we have a μ_2 where $\mu_2 > \mu_1$. Consequently, in some alternate reality we might have had a different μ_2' (we will express it as μ_2' for notational clarity) such that $\mu_2' > \mu_1$. Our information about μ_2 is **not** genie information - it is an assumption that we are making about which reality we exist in. Subsequently, as we examine the performance of our algorithm, we should arrive at some algorithm which does well regardless of the reality which we exist in. The performance of a good algorithm should be impervious to our assumption about our reality.

Thinking about an alternative reality is useful as it underlines the main problem: we need to sample Arm 2 enough times to try and narrow down which reality we exist. We also want to do so with as few samples as possible to avoid pulling the sub-optimal arm too many times (in case we are in the wrong reality and Arm 2 is sub-optimal)! This is a clear instance of the Exploration/Exploitation trade-off that we have seen in past discussions.

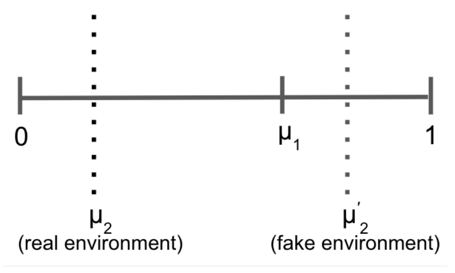

Figure 18.1: Two-armed Bernoulli bandit problem setup

Now, we will introduce some additional notation to help separate the two probability spaces that we are dealing with. The first probability space is the one where $\mu_2 < \mu_1$ and the second probability space is the one where $\mu_1 < \mu_2'$. These two spaces will assign different weights to different events and will therefore should be expressed within the use of different notation. Consider the probability of $010\dots$ appearing when I pull Arm 2. It will be different depending on whether we are operating in the real environment or the fake environment.

Henceforth, we will use P, E for the probability and expectation of events in the real environment where $\mu_2 < \mu_1$. We will use P', E' for the probability and expectation of events occurring in the fake environment, where $\mu_1 < \mu_2'$. All other notation remains consistent with previous lectures.

Everything that follows is a series of steps that we employ in order to calculate the probability of messing up and believing that we exist in a reality other than the one that we exist in. Examining this value helps us lower bound the Expected number of times we will pull the correct arm within the context of different algorithms.

18.3 Explore then Exploit: (Stupid UCB)

In this section we will explore the performance of Explore then Exploit (the Stupid UCB algorithm) within the problem framework which we had constructed in the section above. The main characteristic of the Explore-then-Exploit is that it decouples the exploration and exploitation into two distinct phases: one for exploration and one for exploitation as shown below. Since we already know μ_1, we only need to arrive at an estimate of μ_2 to infer which arm has a better payoff. In turn, we can choose to exploit the arm which has a better payoff all the way till the end of our allotted time (T). The algorithm is stated below.

Algorithm 1 Explore then Exploit

```
1: Exploration Phase: Sample arm 2 for some $T_0$ steps. Observe rewards $X_{2,1} \dots X_{2,t_0}$.
2: Exploitation Phase:
3:     Compute estimate $\hat{\mu_2}(T_0)$.
4:     if $\hat{\mu_2}(T_0) > \mu_1$ then:
5:         Sample arm 2 for the remaining period of time
6:     else:
7:         Sample arm 1 for the remaining period of time
8:
```

To examine the probability of making an error - a good place to start would be to examine the expected number of times we pull the wrong arm in the **fake environment**. Recall that in the fake environment, Arm 2 is optimal. Therefore, we want to examine the Expected number of times we would pull Arm 1 within the fake environment: ($E'[T_1(T)]$).

A good algorithm will perform well regardless of its circumstances. It should work well in the our assumed environment AND it should also work well in any fake/alternate environment that it could be subjected to. If there is an alternate environment in which we mess up really badly then we cannot claim to have an algorithm that works well in general - as our algorithm does not have a consistent performance across all conceivable situations. Therefore, if we are somehow pulling Arm 1 very aggressively by exploiting our problem structure within the real environment - this would make our performance within the fake environment suffer. This would cause the algorithm to pull Arm 1 in the fake environment (as it is inherently biased towards Arm 1) and the expected number of times that we pull the **wrong** Arm (Arm 1) in the fake environment to be unreasonably large. By lower bounding this value we can ensure that it does not grow unreasonably large.

As mentioned in lecture: we could just go ahead and lower bound the probability of messing up our prediction in the real environment as well. It would cause us to arrive at a similar result. It might not, however, ensure us that we are operating with sufficient distance from the information provided within the problem setup.

Let us examine the quantities at hand,

$$E'[T_1(T)] = P'(\hat{\mu_2}(T_0) < \mu_1)(T - T_0) + P'(\hat{\mu_2}(T_0) \geq \mu_1) * 0 = P'(\hat{\mu_2}(T_0) < \mu_1)(T - T_0)$$

$T_1(T)$ is a random variable denoting the number of times we pull arm 1 in a game of T rounds. The quantity is random as it depends on the value that our estimator $\hat{\mu_2}(T_0)$ will take. In turn, this is dependent on the observations of rewards - which is random.

You might recognize the term $P(\hat{\mu_2}(T_0) < \mu_1)$ - this is something that we had Upper Bounded in a previous lecture. Deriving an Upper Bound for this term had allowed us to essentially state how many times we would pull a sub-optimal arm in some environment at most. In turn, we were able to formulate an Upper Bound on Regret. Now we will examine $P'(\hat{\mu_2}(T_0) < \mu_1)$ a little more closely in an attempt to Lower Bound it.

Step 1: Notice that $P'(\hat{\mu_2}(T_0) < \mu_1) = P'(\hat{\mu_2}(T_0) \in [0, \mu_1])$.

By definition itself, we have: $\mu_2 \in [0, \mu_1)$. Furthermore, this implies that $P'(\hat{\mu_2}(T_0) < \mu_1) \geq P(\hat{\mu_2}(T_0) \leq \mu_2)$ as $\mu_2 < \mu_1$ within our problem framework.

Step 2: Recall from the last lecture:

$$P(\hat{\mu_2}(T_0) = \mu_2) \geq \frac{1}{\sqrt{T_0}} e^{-T_0 \times D_{KL}(\mu_2 || \mu_2')}$$

Step 3: We merge these two steps together to get:

$$E'[T_1(T)] \geq \frac{1}{\sqrt{T_0}} e^{-T_0 \times D_{KL}(\mu_2 || \mu_2')}(T - T_0)$$

The term on the right represents how likely we are to confuse the fake environment for the real environment - thereby under sampling the correct arm. If T_0 was a constant then the event above be occurring with a constant probability. This is not a great situation for us to be in - as we want our initial sampling period to scale relative to the time horizon. Therefore, we set our initial sampling period to be scale such that it is of the order $< ln(T)$. In fact, T_0 **must** scale like $ln(T)$ for us to have a reasonable lower bound.

One should notice here we want T_0 to be large enough that we are able to determine - definitively - which environment we exist in. If we have only sampled a small number of times then there would be inherent uncertainty about which environment we could possibly exist in. Figure 18.2 provides some intuition as to why that might be the case. Sampling arm 2 more decreases the overlap between the binary sequences of size T_0 that would appear likely under both environments. Hence as we sample more, we are less and less likely to end up with sequence that might lead us to commit to the wrong arm.

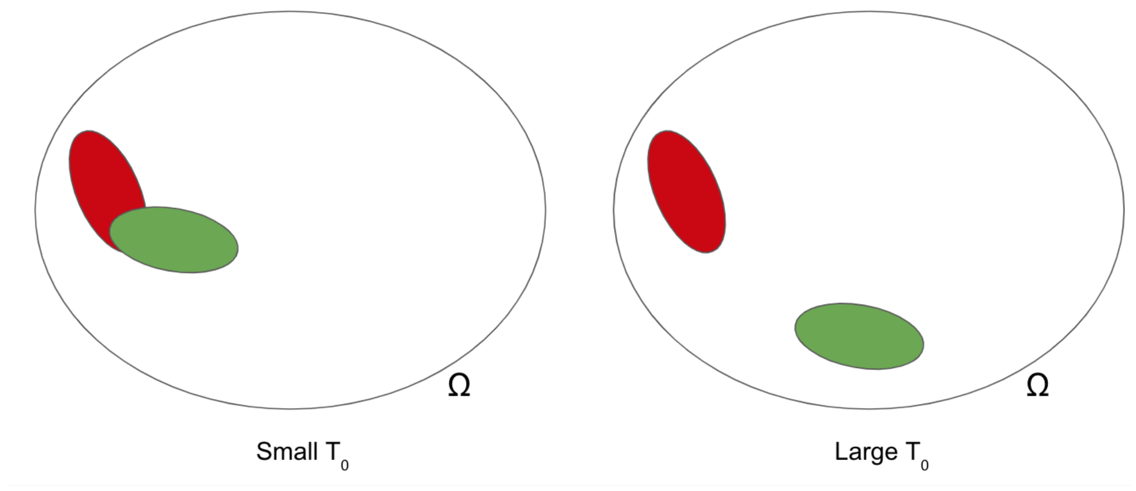

Figure 18.2: $\Omega : \{0,1\}^{T_0}$
Green blob: likely sequences under true environment
Red blob: Likely sequences under fake environment

Essentially, we want to hedge our bets across all possible environments. We do not want to have some algorithm which only works well in the case that Arm 2 is worse Arm 1. The derivation and the steps above establish a sub-case within the space of all algorithms - we have proved a lower bound for the case of the Explore then Exploit category of algorithms.

18.4 Lower Bounds for General Algorithms

So far, we have seen performance guarantees on the regret for algorithms like UCB and Explore then Exploit (Stupid UCB). In our pursuit of better algorithms we are interested in answering the following central questions: How fundamental is the $O(\sqrt{T})$ regret bound we derived for UCB? Is there a fundamental limitation on how well any algorithm can do?

18.4.1 Thinking about general algorithms Stochastic Bandits

In this section, we will try to prove some statement about the minimum regret that any algorithm A must take when solving the problem. We will think of an algorithm A as follows: At every round of the game T_i, the algorithm has to make a decision whether to sample arm 2 more or commit to arm 1 for the rest of the game. We will denote the event of committing to arm 1 after T_i rounds as $E_2[T_i]$. This is how we will choose to parameterize any possible algorithm that solves our problem. Notice that although the event is defined the same way for any algorithm, the decision rule that determines whether the algorithm will explore or exploit is specific to the algorithm we choose. You can actually think of whether the event happens or not as a function specific to A that observes $X_{2,1} \ldots X_{2,T_0}$, the observations from our environment up to time T_0.

Aside: The vending machine view

Thinking about a general algorithm as being defined by its decision rule of when to abandon exploring and commit to a hand is still a little restrictive. One could imagine that we have an algorithm (let's call this algorithm A) that could instead interleave exploration and exploitation steps before making a final decision to commit to an arm.

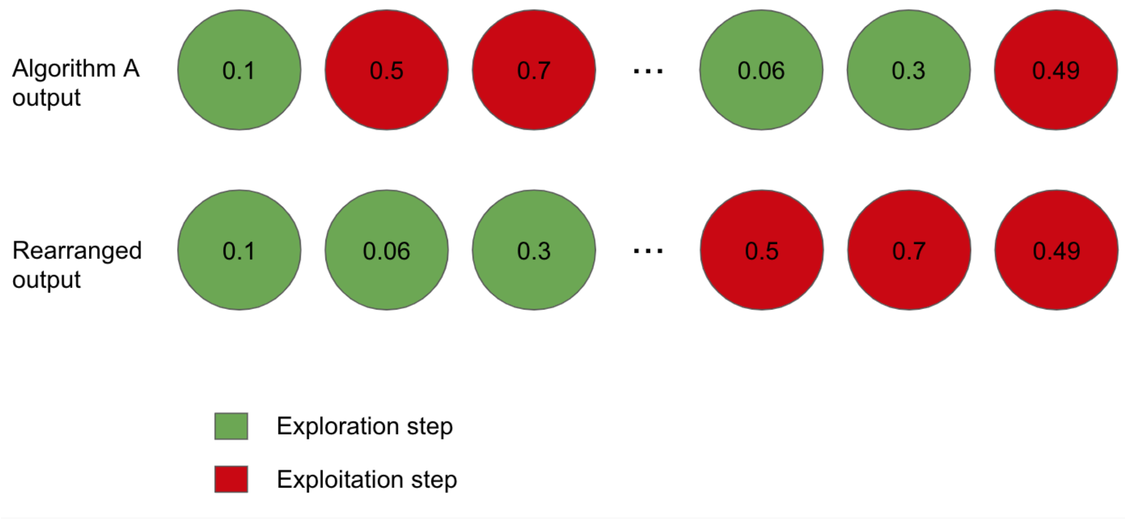

Figure 18.3: What if an algorithm interleaves exploration and exploitation steps?

We will now try to give some intuition as to why such an algorithm can still be analyzed within the framework discussed above. Consider an Algorithm A that interleaves exploitation and exploration steps before committing to an arm. It is hence not entirely unreasonable that the algorithm's behavior looks a little like the first row of figure 18.3. Note that the probability of observing the specific sequence is the same as the probability of observing a sequence where I have rearranged the time-steps such that the exploration steps all proceed the exploitation steps. The reason for that results at each time-step are i.i.d Bernoulli. Since the probability of the observed sequence and the total reward remain unchanged under permutation of the steps, we can think of Algorithm A as having an equivalent algorithm that explores first, then commits to an arm.

This presents an alternative view of generating sequences given an algorithm and an environment which most resembles a simple vending machine. Imagine that instead of generating a sequence dynamically by having your algorithm pick an arm at every step, you instead have access to a vending machine that, given: a) your algorithm and b) the true environment parameters μ_1 and μ_2, samples a reward sequence (with tags that tell you whether each step was an exploration step or an exploitation step) and returns it to you. To analyze the algorithm all you have to do is rearrange the sequence as described above to cast the algorithm in the framework we discussed.

In the following sections, we will be considering algorithms such that $E[T_{worse}(T)] = o(T^\delta)$ where $\delta > 0$. In this case T is the total number of rounds played and $T_{worse}(T)$ is the number of times we pull the worse of the two arms in T rounds of the game. This essentially means that we will not be considering algorithms that experience linear regret at each round but rather those whose regret at each round becomes vanishingly small as the number of rounds increases.

18.4.2 Thinking about alternative universes

As we are playing the game, the only information we have at our disposal is the value of μ_1 and the reward realizations that we have observed so far. At any point in the game, we can have multiple different values of μ_2 that would be reasonable enough given the rewards we have observed with each of these possible values corresponding to a different universe. Clearly, as we sample arm 2 more, we will be able to narrow down more effectively the set of possible universes we could be in. We know that $Regret \approx T \times P(\text{we are wrong})$. In fact we want to our algorithm to be doing well in any of the candidate universes that are likely given the observed rewards. This can be expressed mathematically as

$$Regret \approx T \times \max_{u \in U} P(we\ are\ wrong\ in\ u) \times P(we\ are\ in\ u)$$

To get a lower bound on regret, we want to lower bound the probability of making an error in any universe we might be in. A lower bound on regret would therefore tell us how good the best algorithm we could devise can be. In our analysis, we will consider a simplified scenario where we are trying to ensure our algorithm performs reasonably well in two universes: one real and one fake (but likely given the data we have seen so far). A generalization of this idea to multiple environments is also possible.

In the next section, we will try to show:

Theorem 18.1 *For any algorithm A,*

$$\frac{E[T_{worse}(T)]}{ln(T)} \gtrapprox \frac{1}{D_{KL}(\mu_2||\mu_1)}$$

18.4.3 Analysis

For the purposes of this analysis, we will (somewhat arbitrarily) define the quantity $f_T = \frac{ln(T)}{D_{KL}(\mu_2||\mu_2')}$ to be the number of times we will explore arm 2 before making our final decisions about which arm to pull in the context of a T-round game.

We begin our analysis by considering the probability $P(T_2(T) < f_T)$ which essentially measures the probability that, in the real environment, we will commit to arm 1 (the better hand) before round f_T. Note that we can relate this probability to the events $E_2(T)$ that we described in the previous sections. $P(T_2(T) < f_T) = \sum_{i=1}^{f_T-1} P[E_2(T_0)]$. This probability is the union of disjoint events that either the algorithm stops sampling at round 1 or it stops at round 2 and so on and so fourth.

Now define the event $E^* = T_2(T) < f_T$. Note that if we are in the real universe E^* is clearly a desirable event since we rule out the worse hand and we would like it to happen as soon as possible while if we are in the alternate universe, E^* would denote the event that we make a mistake. Our goal is to relate $P(E^*)$ to $P'(E^*)$. Clearly, there is a balance we need to strike. In the real environment, ruling out arm 2 early on is clearly desirable while in the alternative environment, ruling arm 2 out early is not desirable at all. We want an algorithm that operates well in both scenarios since in practice, we will never actually know which environment we will be in.

How should $P(E^*)$ and $P'(E^*)$ relate to each other?

Suppose T_0 is large, then the two probabilities is quite different because a large number of samples can help us figure out which environment we are in. Low T_0 does not let us distinguish between the two environments so in our heads the probability of ruling arm 2 out is the same in both cases since in our head, we are still confused as to which environment we are actually in.

So now lets connect the two probabilities in an approximate sense.

$$P'(E^*) \approx P(E^*)e^{-f_T D_{KL}(\mu_2||\mu_2')}$$

Now lets see if this matches our intuition. We see that $P'(E^*)$ will be substantially different to $P(E^*)$ if the quantity in the exponent is really large or really small. WLOG lets consider that μ_2 and μ_2' are chosen such that the KL divergence is always non-negative. Then, if we fix the KL divergence term between the two, increasing f_T (the number of times we sample) will indeed make the two probabilities more different while a smaller f_T will send the exponent to 0 hence making the two probabilities almost equal. This makes sense since the more you sample, the easier should be to distinguish between the environments which should introduce this asymmetry in the probabilities. On the contrary, fixing the number of times we sample and changing the divergence term will make the probabilities more equal if it the universes are fundamentally very similar while it will make the probabilities more different if the environments are fundamentally different (hence the same number of exploration steps is enough to make the distinction).

Now, lets check the implications that sampling arm 2 might have on the the regret term. By plugging $f_T = \frac{(1-\epsilon)\log T}{D_{KL}(\mu_2||\mu_2')}$ we get that $P'[E^*] \geq P[E^*]e^{-(1-\epsilon)\log T} \geq \frac{1}{T^{(1-\epsilon)}}$ which implies that the loss in the fake environment $\geq \frac{1}{T^{(1-\epsilon)}}$. This implies that $Regret \geq \frac{1}{T^{(1-\epsilon)}} \times T = T^{\epsilon}$.

Conclusion

In this section, we have proved a very important fact, namely that the explore/exploit tradeoff we discussed in the context of UCB is not specific to UCB itself but rather to the class of problems that we are trying to solve as a whole.

18.5 Bayesian Bandits and Thompson Sampling

So far, everything we have considered has been in the frequentist realm of things: we have observed rewards in order construct point estimates which dictate our course of action. It is natural for one to wonder what would happen in the case where we had higher order information about μ_1 and μ_2 available to us and how would would go about constructing an algorithm that takes advantage of such higher order information.

Once again, we will consider a simple 2-armed setup where at each time-step, rewards are drawn from a $Ber(\mu_1)$ for arm 1 and a $Ber(\mu_2)$ from arm 2. However, in this case we will think of parameters as coming from some underlying Prior distributions over $[0, 1]$, say $\mu_1 \sim P_1$ and $\mu_2 \sim P_2$.

This creates potential for two interesting scenarios. The first is wherein we can consider $\hat{q_1} = P(\mu_1 > \mu_2)$ (and similarly $\hat{q_2}$) the probability that Arm 1s reward is higher than the reward for Arm 2. Knowing the Prior distributions over Arm 1 and Arm 2 - gives us prior information about which Arm is better through $\hat{q_1}, \hat{q_2}$.

Another assumption we will revise in Bayesian Bandits paradigm has to do with the fact that pulling Arm 1 tells us nothing about Arm 2. In fact, we could have a joint distribution over μ_1 and μ_2 in which case by sampling Arm 2, we automatically collect some information about Arm 1 as well.

As you can see, this is a richer problem structure and allows us to incorporate more nuance by making certain assumptions. Having useful priors or having dependence between arms allow us to gain an algorithmic edge over the previous problem framework.

Notation:

I_t = the arm selected at time t in the game.

$X_{I_t,t}$ = The reward we observed from selecting arm I_t at time t.

$\hat{q_1}(t-1)$ = The probability distribution that $\mu_1 > \mu_2$ given that we have observed rewards $(X_{I_1,1}, X_{I_2,2}, X_{I_3,3}, \ldots, X_{I_{t-1},t-1})$ so far. The updates are p is simply Bayes Rule. Assume that we have played the game for $t-1$ rounds and we want to see what to do on round t. A very simple strategy is described below.

At every round of the game, we will simply maintain updated distributions $\hat{q}_1(t-1)$ and $\hat{q}_2(t-1)$ and choose the next arm randomly based on the following rule.

I_t = arm 1 with probability $\hat{q}_1(t-1)$ or arm 2 with probability $\hat{q}_2(t-1)$

We must bear in mind certain dynamics such as the imbalance between the number of times we sample Arm 1 versus the number of times we sample Arm 2. This would make certain posterior information more reliable than other posterior information.

The approach above is known as Thompson Sampling - and it works very well. It might make you a little uncomfortable to notice that there is no hyper-parameter within this entire algorithmic pipeline. There is no degree of explicit parametric control over this algorithm. As we will see in subsequent lectures, we are very much in implicit control of this algorithm.

Bayesian Bandits, Thompson Sampling and similar ideas will be analyzed in greater depth over the subsequent lectures. For an interesting blog post which examines Bayesian Bandits, check out: https://www.chrisstucchio.com/blog/2013/bayesian$_{b}andit.html$.

References

1. S. Bubeck and N. Cesa-Bianchi, Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems, Foundation and Trends in Machine Learning, vol 5, no 1, pp 1-122, 2012.

2. Andherbertrobbins, T L Lai. Asymptotically Efficient Adaptive Allocation Rules.