Reinforcement Learning

Jacob Denson

December 3, 2016

Table Of Contents

Part I

Bandits

Reinforcement learning attempts to design intelligences that can learn from its own interactions with an environment. It differs from standard machine learning techniques in that it must choose how to interact with the environment to obtain data to learn from. In its most general form, an agent must maximize a stochastic reward function in response to changes in its own environment. The main idea of reinforcement learning is the reward hypothesis – that all goals can be thought of as the maximization of some reward signal corresponding to actions that an agent performs. The main path to intelligence is then to find strategies to maximize this reward signal. There is perhaps some concern with this view, in that 'real intelligence' does not have some reward function; endorphins in the brain provide some counterexample to this argument. We shall begin by discussing the most mathematically tractable machine learning problem – the N armed bandit, and then move on to more challenging situations.

Chapter 1

Bandits 101

Imagine you are at a casino, with rows upon rows of slot machines. A strategic game player would quickly determine the slot machine that maximizes your profit, and exploits this machine to obtain the best payoff. From the start, you have no idea of the mechanisms determining how the rewards are given. Thus you must balance the act of determining these mechanisms by trial and error with exploiting the knowledge you have obtained. The problem of maximizing the amount of money you make is known as the **multi armed bandit problem**.

In the general reinforcement learning problem, the exploration exploitation tradeoff mentioned in the last paragraph is just as prevelant. However, the bandit problem is much more mathematically tractable, and is therefore more prone to theoretical analysis. The problem is not just a toy problem though. In medical studies, we want to obtain information about the potency of medicine as early as possible, to prevent possible harm to those in the study – this was the original context in which the bandit problem was introduced, by William R. Thompson in 1933. In the 1950s, psychologists designed an experiment which placed mice in a T-shaped maze, seeing if they could learn to consistently choose a direction in order to get food. To study a similar problem in humans, the scientists imagined a two armed slot machine, where pulling each arm gives different distributions. In a test where questions are sequentially generated, one wants to ask questions which best evaluate a student's skill. That these Problems can be solved by mathematical means is a windfall to those who need reliable solutions to the related problems.

Mathematically, the domain of the bandit problem is based on a finite set of actions A. Then, for each $a \in A$, we fix a reward distribution $\rho(a)$ over the real numbers. The set of all collections of such distributions will be denoted $\mathcal{E}(A)$, which we call the space of all **environments** over the action set A. Given a fixed environment, the goal is to find a sequence of actions to maximize our reward, subject to the fact that we don't know our environment in advance. We must learn optimal actions by choosing exploratory moves in such a way that we are minimally penalized.

A policy is the mathematical model of a strategy which, given a memory of past actions, decides (possibly randomly) on the next action to take. Deterministic policies are simplest to define. First, define a **history** of length k to be an element of $(A \times \mathbf{R})^k$ – a sequence of data obtained from interactions in the environment. We shall denote the set of all histories of length k by $\mathcal{H}_k(A)$. We shall let $\mathcal{H}_0(A)$ consist of the single empty tuple. A **decision function** for a policy is a sequence of functions π_i, which map histories in $\mathcal{H}_k(A)$ to probability distributions over A which decide based on prior experience what to take as an i'th action. The decision function induces a random sequence of actions $A_1, A_2, \dots$, rewards $X_1, X_2, \dots$, and histories $H_k = [(A_1, X_1), \dots, (A_k, X_k)]$, such that

$$A_k = \pi_k(H_{k-1}) \qquad (X_k | H_{k-1}, A_k) \sim \rho(A_k)$$

More generally, a **stochastic policy** maps historys to probability distributions over A. A stochastic policy then induces a sequence $A_1, A_2, \dots$ and rewards $X_1, X_2, \dots$, together with a filtration $\{\mathcal{F}_1, \mathcal{F}_2, \dots\}$ such that A_k is $\mathcal{F}_k$-measurable, X_k is $\mathcal{F}_{k+1}$-measurable, and

$$(X_k | \mathcal{F}_k) \sim \rho(A_k)$$

The goal of the bandit problem is to choose a policy which induces the maximum expected reward

$$Q_n = \sum_{k=1}^{n} X_k$$

for some value $n \in \mathbf{N}$ (the 'finite horizon problem') or to optimize the order of growth of the expected reward as $n \to \infty$ (the infinite horizon problem). Sometimes we may not know n, and we will develop so-called **anytime algorithms** which give good results without knowledge of the horizon length.

For each action $a \in A$, let

$$\mu(a) = \mathbf{E}(X_k | A_k = a) = \int_{\mathbf{R}} x d\rho(a)(x)$$

We then define $\mu^* = \max \mu^a$, which is the reward taken for choosing the optimal action. If a user knew the distributions of each action, he would obviously try and pick the action corresponding to μ^* every time. The real problem is learning if μ^* is optimal, while still optimizing the reward obtained. This is the *explore-exploit tradeoff*; if we wish to maximize reward, we must attempt to use the arms which have the highest estimated mean reward. But if we only exploit, then we will never learn the optimum reward, because we need to sample all the arms to obtain good estimates of the mean reward, so we need to 'explore' as well. Good general-use policies must balance exploring and exploiting.

1.1 The Regret Function

Another way to visualize the problem is that rather than maximizing 'good' decisions, we minimize the number of 'bad' decisions we make. Define the regret function

$$R_n = \mathbf{E}\left[\sum_{k=1}^{n} (\mu^* - X_k)\right]$$

Maximizing the expected reward in a particular environment of the bandit problem is exactly the same as minimizing the regret. Indeed, we may write

$$R_n = n\mu^* - Q_n$$

and thus the R_n is obtained from Q_n by a sign change and a shift, provided we are only working over a single environment. However, the regret is translation invariant, making it a much better measure of an algorithm's learning rate if we analyze asymptotics of a policy over many environments. One drawback is that the regret is not scale invariant, which means we normally assume the means of the rewards are bounded an interval, usually the unit interval $[0, 1]$.

To calculate the regret, it is often useful to apply a certain formula measuring the expected regret in terms of the number of bad decisions we make. Define a new random variable

$$T_n(a) = \#\{1 \leqslant k \leqslant n : A_k = a\}$$

which counts the number of times the action a is chosen. We find that

$$\begin{aligned}
\mathbf{E}\left[\sum_{k=1}^{n}(\mu^* - X_k)\right] &= \sum_{k=1}^{n}\mu^* - \mathbf{E}[\mathbf{E}[X_k|A_k]] \\
&= \sum_{a \in A}(\mu^* - \mu_a)\sum_{k=1}^{n}\mathbf{P}(A_k = a) \\
&= \sum_{a \in A}(\mu^* - \mu_a)\mathbf{E}(T_n(a)) \\
&= \sum_{a \in A}\Delta_a \mathbf{E}(T_n(a))
\end{aligned}$$

where $\Delta_a = \mu^* - \mu_a$ is the **action gap** between a and the optimal action.

In applied statistics, it is normal to blindly apply the law of large numbers to conclude that, provided we sample an action infinitely many times, the random variables

$$\widehat{\mu_n}(a) = \frac{1}{T_n(a)}\sum_{k=1}^{n}\mathbf{I}\{A_k = a\}X_k$$

converge to the sample mean. However, for our purposes, we want to obtain explicit bounds on the quantities obtained, so the convergence rate of the sample means comes into question. In general, the convergence rate is questionable, but reasonable assumptions about random variables will give us quantifiable bounds which we can use to bound the regret we obtain.

One problem with calculating the means above is that the naive way needs to store all $T_n(a)$, $\mathbf{I}(A_k = a)$, and X_n. The formula above allows to calculate the sample means in $\Theta(n^2|A|)$ time, with $\Theta(n|A|)$ memory usage. However, there is a computational trick for keeping track of these estimates without having to store the values of the indicator function for all k. Write

$$\widehat{\mu_{n+1}}(a) = \frac{T_n(a)\widehat{\mu_n}(a) + \mathbf{I}\{A_{n+1} = a\}X_{n+1}}{T_{n+1}(a)}$$

and this gives a recurrence relation meaning we need only keep track of $T_n(a)$ and $\widehat{\mu_n}(a)$, for the rest of the data is given to us sequentially and therefore does not need to be memorized. Thus we can calculate new sample means in $\Theta(|A|)$ time, and we need only memorize the $T_n(a)$ for the most recent n, so that we only use $\Theta(|A|)$ memory.

1.2 Naive and Epsilon-Greedy Policies

A naive policy, the greedy method, always takes the action with the highest sample average. The problem with this method is, if the sample reward for some action becomes distorted from it's actual mean, it may never be taken again, even if the actual expected reward is much higher. This normally occurs if the variance of actions is too high. Given a low-variance reward function, the greedy policy may choose the right action, but it is very difficult to give a mathematical guarantee. One helpful trick may be to set $\widehat{\mu_0}(a)$ optimistically, that is, not the default value of zero but some value higher than all rewards for an action. Then all values are at least tried once before they begin to settle down, so some exploration is done. Again, we cannot prove anything successful about this policy because it's decisions are so volatile. A somewhat smarter policy, the ε-greedy method, attempts to fix this issue by guaranteeing convergence at the cost of the hope of complete optimality.

Fix some real number ε with $0 \leqslant \varepsilon \leqslant 1$. The idea of the ε greedy method is to act greedily, except for a small portion of time where we act randomly, ensuring we sample enough times for our sample averages to converge. The ε greedy policy π is defined for each action a by the distribution

$$\mathbf{P}(A_{n+1} = a | H_n) = \begin{cases} \varepsilon + \frac{1-\varepsilon}{|A|} & : a = \text{argmax}_{b \in A}\ \widehat{\mu_b}(n) \\ \frac{1-\varepsilon}{|A|} & : \text{otherwise} \end{cases}$$

We can use the regret decomposition formula to conclude that

$$R_n = \sum_a \Delta_a \mathbf{E}(T_n(a)) \leqslant n \sum_a \Delta_a \left(\varepsilon + \frac{1-\varepsilon}{|A|} \right)$$

so that we have $O_{\varepsilon,\Delta}(n)$ expected regret; this is not the best situation to be in, but a good start, for at least we have some mathematical bound. We also have upper bounds

$$R_n \geqslant \Delta_a \mathbf{E}(T_n(a)) \geqslant \Delta_a \frac{1-\varepsilon}{|A|}$$

so R_n is in fact $\Theta_{\varepsilon,\Delta}(n)$.

So what are the shortcomings of the ε-greedy policy? The first is that it considers every action by a probability that is lower bounded by a constant greater than zero, so that we have guaranteed linear regret. In order to obtain better results, we must choose actions with probabilities which vary over time – being large over all actions initially to explore our opportunities, then honing in on potential good choices.

1.3 Explore Then Commit

Another intuitive strategy which seems viable is to explore for a certain amount of time, and once this time is finished, decide on an optimal action and then always perform that action. A mathematical analysis of this strategy requires that the variance of the function by well-defined, so that we assume the reward distribution are always subgaussian (1-subgaussian to make the formulas nice, but we the results can be applied to all subgaussian variables by scaling).

The explore then commit strategy is defined with respect to a parameter m, which samples each action m times equally, obtaining estimates $\widehat{\mu}_a$ of the means. The policy then chooses the action a' corresponding to the maximum estimated mean $\widehat{\mu}_a$. The rate of growth of the expected regret then becomes the difference

$$\sum_a \mathbf{P}(a' = a)\Delta_a$$

If a^* is an optimal action, then

$$\begin{aligned}\mathbf{P}(a' = a) &\leqslant \mathbf{P}(\widehat{\mu}_a - \widehat{\mu_{a^*}} \geqslant 0)\\ &= \mathbf{P}((\widehat{\mu}_a - \mu_a) + (\mu^* - \widehat{\mu_{a^*}}) \geqslant \Delta_a)\end{aligned}$$

so determining the optimality of the algorithm depends on properties of the tail distribution of the random variable $Y = (\widehat{\mu}_a - \mu_a) + (\mu^* - \widehat{\mu_{a^*}})$. Note that if the reward distributions are 1-subgaussian, then Y is $2/m$ subgaussian, and so by standard properties of subgaussian random variables, we find

$$\mathbf{P}(Y \geqslant \Delta_a) \leqslant e^{-m\Delta_a^2/4}$$

and so we can bound for the expected regret to be

$$R_n \leqslant \sum_a \Delta_a \left[m + (n - |A|m)e^{-m\Delta_a^2/4} \right]$$

The expected regret per round can be very large for small values of m, but decreases exponentially as m is increased. However, as we increase m the total regret obtained over the initial rounds increases.

For a two-armed bandit, the expected regret after n rounds (for large enough m) will be bounded by

$$m\Delta + (n - 2m)\Delta e^{-m\Delta^2/4} \leqslant m\Delta + n\Delta e^{-m\Delta^2/4}$$

We should expect these bounds to be tight, the first because the subgaussian bound is tight, and the second because if m is large, $e^{-m\Delta^2/4}$ will be very small, and the difference between $(n-2m)e^{-m\Delta^2/4}$ and $ne^{-m\Delta^2/4}$.

If we knew Δ and n in advance, we could minimize this bound by choosing the optimal m. Indeed, we find that

$$\frac{\partial}{\partial m}(m + ne^{-m\Delta^2/4}) = 1 - (\Delta^2 n/4)e^{-m\Delta^2/4}$$

and setting this equation equal to zero gives us

$$m = \frac{4}{\Delta^2} \log\left(\frac{n\Delta^2}{4}\right)$$

so taking the actual exploration number to be the floor or ceiling of this number should give us optimal estimates, giving us a regret bound approximately equal to

$$R_n \leqslant \frac{4}{\Delta}(1 + \log(n\Delta^2/4))$$

If $\Delta \to 0$, this bound becomes useless. However, we can always use the trivial bound $R_n \leqslant n\Delta$, so that

$$R_n \leqslant \min\left(n\Delta, \frac{4}{\Delta}(1 + \log(n\Delta^2/4))\right)$$

The only problem with this strategy is that we require knowledge of both n and Δ to obtain logarithmic regret, but this regret depends on both Δ and n, and we rarely know n ahead of time, let alone Δ_a ahead of time in real applications.

Suppose we know m, but not Δ. The mathematical way to measure how good a policy is if it doesn't know some information is to consider the worst case regret with respect to Δ. If we let Δ take all positive values, then this worst-case would always be infinite, so we normally enforce that $\Delta \in [0,1]$, and so we want to optimize

$$\max_{\Delta \in [0,1]} \min_{1 \leqslant m \leqslant n} R_n(m,\Delta)$$

where $R_n(m,\Delta)$ is the expected regret over the two-armed bandit environment corresponding to Δ, and m is the exploration parameter to the explore then commit algorithm. We shall prove asymptotically tight bounds on the 'correct' choice of m in this situation.

If $\Delta \leqslant 4n^{-1/3}$, then we trivially have a bound

$$R_n(m,\Delta) \leqslant n\Delta \leqslant 4n^{2/3}$$

regardless of our choice of m. If we assume $\Delta \geqslant 4n^{-1/3}$, and if we choose $m = \log(n)n^{1/3}/3$, we find

$$\begin{aligned} R_n(m,\Delta) &\leqslant \Delta(m + ne^{-m\Delta^2/4}) \\ &= \Delta(\log(n)n^{1/3}/3 + ne^{-\log(n^{1/3})}) \\ &= \Delta(\log(n)n^{1/3}/3 + n^{2/3}) = \Delta O(n^{2/3}) \end{aligned}$$

So we have a $\Delta O(n^{2/3})$ bound for the regret. This is effectively the best we can do. For any n and m, if we let $\Delta^2 = 4\log(n^{1/3})/m$, then we find

$$R_n(m,\Delta) \approx \Delta(m + ne^{-m\Delta^2/4}) = \Delta(m + n^{2/3}) \geqslant \Delta n^{2/3} = \Delta\Omega(n^{2/3})$$

This asymptotic lower bound still occurs if we assume the action gap is bounded. If our choice of Δ above happens to be > 1, then we conclude $\log(n) > 3m/4$, which means we hardly explore at all. We then find

$$\Delta(m + ne^{-m\Delta^2/4}) > \Delta ne^{-\log(n)\Delta^2/3} = \Delta n^{1-\Delta^2/3}$$

Choosing $\Delta = 1$ gives us a $\Delta n^{2/3}$ regret lower bound (since we hardly explore, a large action gap has little effect on the power of the algorithm determining the best action). There are algorithms which have $O(n^{1/2})$ regret, so the explore then commit bandit is still asymptotically suboptimal.

1.4 The UCB Algorithm

A more advanced and recent solution to the problem results from acting optimistically – we always choose our actions as if the environment was as nice as plausibly possible. In this case, we either choose the best action (our optimism was deserved), or we choose a suboptimal action which we learn is worse than others available. The difference between this algorithm and the greedy algorithm is that confidence intervals are much more tractable to formally analyze, and it's easier to prevent yourself from becoming 'trapped' in a suboptimal action. If $X_1, \dots, X_n$ are i.i.d and 1-subgaussian, then the sample means satisfy

$$\mathbf{P}(\widehat{\mu_n} \geqslant \varepsilon) \leqslant e^{-n\varepsilon^2/2}$$

By a change in variables, we find

$$\mathbf{P}\left(\widehat{\mu_n} \geqslant \sqrt{\frac{2}{n}\log(1/\delta)}\right) \leqslant \delta$$

So we have constructed confidence intervals with a probability of δ being correct. If we are optimistic, then it is consistent with our hypothesis that

$$\mu = \widehat{\mu_n} + \sqrt{\frac{2}{n}\log(1/\delta)}$$

The upper confidence bound algorithm chooses the action a which maximizes this quantity (after choosing each action once, so that the values are well defined). We see now that our algorithm is more optimistic for small values of δ, and less optimistic for large values. It will be useful to decrease δ over time, to avoid becoming 'trapped' in suboptimal actions, or to become more and more confident in our action over time.

A standard choice is $\delta_n = 1 + n\log^2 n$, which increases slightly faster than n. Why is this chosen? If an optimal arm is discredited even once, an algorithm must pay linear regret. If we fix δ, we will choose a suboptimal action $(1-\delta)$ of the time. Trying to fix this by adjusting the δ so that this occurs with $1/n$ probability, we find that we need to choose δ_n.

If our algorithm is to achieve good regret bounds, it should only pick an arm many times if it is highly confident of the arm's reward. This leads directly to the notion of confidence intervals, and thus to the upper confidence bound algorithm. The upper confidence bound algorithm is asymptotically much better than the other algorithms we have considered. While all other algorithms have linear regret, this algorithm actually has logarithmic regret. This is proven by analyzing the number of times a suboptimal action a is taken. First, a lemma.

Lemma 1.1. *Let $X_1, X_2, \dots$ be a sequence of 1-subgaussian random variables, with sample average $\widehat{\mu_n}$, $\varepsilon > 0$, and*

$$Y = \sum_{k=1}^{n} \mathbf{I}\left(\widehat{\mu_k} + \sqrt{\frac{2M}{k}} \geqslant \varepsilon\right)$$

Then

$$\mathbf{E}[Y] \leqslant 1 + \frac{2}{\varepsilon^2}\left(M + \sqrt{\pi M} + 1\right)$$

Proof. We just use the one inequality we know for subgaussian variables. Let $u = 2M/\varepsilon^2$. Then

$$\begin{aligned}
\mathbf{E}[Y] &= \sum_{k=1}^{n} \mathbf{P}\left(\widehat{\mu_k} + \sqrt{\frac{2M}{k}} \geqslant \varepsilon\right) \\
&\leqslant (u+1) + \sum_{k=\lceil u+1 \rceil}^{n} e^{-n(\varepsilon - \sqrt{2M/n})^2/2} \\
&\leqslant (u+1) + \int_{u}^{\infty} e^{-t(\varepsilon - \sqrt{2M/t})^2/2} dt \\
&= 1 + \frac{2}{\varepsilon^2}(M + \sqrt{\pi M} + 1)
\end{aligned}$$

and this gives us the needed bound. □

To analyze the regret of the UCB algorithm, we shall bound $\mathbf{E}(T_n(a))$, for a non-optimal action a. Note that for a fixed ε,

$$\begin{aligned}
T_n(a) &= \sum_{k=1}^{n} \mathbf{I}(A_k = a) \\
&\leqslant \sum_{k=1}^{n} \mathbf{I}\left(\widehat{\mu^*}(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \leqslant \mu^* - \varepsilon\right) \\
&\quad + \mathbf{I}\left(\widehat{\mu}_a(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \geqslant \mu^* - \varepsilon \text{ and } A_k = a\right)
\end{aligned}$$

Let's bound the first term, using the fact that tails of subgaussian variables are small. If $T_a(k-1) = m$, then $\widehat{\mu}_a$ is $1/m$ subgaussian, and so

$$\begin{aligned}
&\mathbf{E}\left[\sum_{k=1}^{n} \mathbf{I}\left(\widehat{\mu^*}(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \leqslant \mu^* - \varepsilon\right)\right] \\
&= \sum_{k=1}^{n} \mathbf{P}\left(\widehat{\mu^*}(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \leqslant \mu^* - \varepsilon\right) \\
&\leqslant \sum_{k=1}^{n}\sum_{m=1}^{k} \mathbf{P}\left(\widehat{\mu^*} + \sqrt{\frac{2\log 1/\delta_k}{m}} \leqslant \mu^* - \varepsilon \,\middle|\, T_a(k-1) = m\right) \mathbf{P}(T_a(k-1) = m) \\
&\leqslant \sum_{k=1}^{n}\sum_{m=1}^{k} \exp\left(-(m/2)\left(\varepsilon + \sqrt{2\log(1/\delta_k)/m}\right)^2\right) \\
&\leqslant \sum_{k=1}^{n} \frac{1}{\delta_k} \sum_{m=1}^{k} e^{-m\varepsilon^2/2} \\
&\leqslant \sum_{k=1}^{n} \frac{1}{1 + k\log^2 k} \frac{1}{1 - e^{-\varepsilon^2/2}} \\
&\leqslant \frac{2}{1 - e^{-\varepsilon^2/2}} \leqslant \frac{4}{\varepsilon^2}
\end{aligned}$$

where we use the fact that $\delta_k = 1 + k\log^2 k$. We apply the previous lemma to the second indicator function to obtain

$$\begin{aligned}
&\mathbf{E}\left(\sum_{k=1}^{n}\mathbf{I}\left(\widehat{\mu}_a(k-1)+\sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}}\geqslant \mu^*-\varepsilon \text{ and } A_k=a\right)\right)\\
&\leqslant \mathbf{E}\left[\sum_{k=1}^{n}\mathbf{I}(\widehat{\mu}_a(k-1)+\sqrt{\frac{2\log 1/\delta_n}{T_a(k-1)}}\geqslant \mu^*-\varepsilon)\right]\\
&\leqslant \mathbf{E}\left[\sum_{k=1}^{n}\mathbf{I}((\widehat{\mu}_a(k-1)-\mu_a)+\sqrt{\frac{2\log 1/\delta_n}{T_a(k-1)}}\geqslant \Delta_a-\varepsilon)\right]\\
&\leqslant 1+\frac{2}{(\Delta_a-\varepsilon)^2}\left(\log 1/\delta_k+\sqrt{\pi\log 1/\delta_k}+1\right)
\end{aligned}$$

Putting the two bounds together, we obtain

Theorem 1.2. *The regret of UCB is bounded by*

$$R_n \leqslant \sum_{a\neq a^*}\Delta_a \inf_{\varepsilon\in(0,\Delta_a)}\left(1+\frac{5}{\varepsilon^2}+\frac{2}{(\Delta_a-\varepsilon)^2}\left(\log 1/\delta_n+\sqrt{\pi\log 1/\delta_n}+3\right)\right)$$

If we let $\varepsilon=\Delta_a/2$ in each sum, we find

$$R_n \leqslant \sum_{a\neq a^*}\left(\Delta_a+\frac{1}{\Delta_a}\left(8\log 1/\delta_n+8\sqrt{\pi\log 1/\delta_n}+44\right)\right)$$

and there is a universal constant $C>0$ such that for all $n\geqslant 2$,

$$R_n \leqslant \sum_{a\neq a^*}\frac{C\log n}{\Delta_a}$$

If we let $\varepsilon=\log^{1/4}(n)$, and let $n\to\infty$, we find

$$\limsup R_n/\log(n) \leqslant \sum_{a\neq a^*}\frac{2}{\Delta_a}$$

So R_n has logarithmic regret.

We do not need to know the action gaps in order to use UCB, but we require the action gaps in order to obtain the regret bound above. But it is fairly easy to obtain a action-gap free bound. Indeed, if $\Delta = \sqrt{|A| C \log n / n}$, then

$$\begin{aligned} R_n = \sum \Delta_a \mathbf{E}(T_n(a)) &= \sum_{\Delta_a < \Delta} \Delta_a \mathbf{E}(T_n(a)) + \sum_{\Delta_a \geqslant \Delta} \Delta_a \mathbf{E}(T_n(a)) \\ &\leqslant n\Delta + \sum_{\Delta_a \geqslant \Delta} \frac{C \log n}{\Delta_a} \\ &\leqslant n\Delta + |A| \frac{C \log n}{\Delta} = \sqrt{|A| C n \log n} \end{aligned}$$

so that the regret is $O(|A| n \log n)$.

One can improve these bound if we assume the distributions of the arms of the bandit are better than subgaussian. This is mainly done by assuming that the rewards are Bernoulli. Indeed, if we assume $\rho(a)$ is a Bernoulli distribution with mean λ and variance $\lambda(1-\lambda)$. We bound the regret by the two terms bounded above. We keep the same bound for the first one, but for the second bound, we use that ... (I'll finish this later)

Chapter 2

Lower Bounding Regret

It turns out that, if we measure a bandit policy by its worst case regret over all the environments in which the policy makes sense, the UCB algorithm has the best asymptotic bound possible over all situations. To show this, we attempt to find the best possible regret we could obtain, assuming our environment is chosen as the worst possible with respect to the current policy. Thus, we are measuring a policy according to the **minimax regret**. We fix some subset $\mathcal{E} \subset \mathcal{E}(A)$, and then take

$$R_n^*(\mathcal{E}) = \inf_{\pi} \left(\sup_{E \in \mathcal{E}} R_n(\pi, E) \right)$$

The value is independant of any policy or environment, and for any policy π, there is an environment $E \in \mathcal{E}$ such that $R_n(\pi, E) \geqslant R_n^*(\mathcal{E})$. It essentially tells us how difficult a particular class of environments is for a policy to optimize over. Thus lower bounding R_n^* is exactly what we need to lower bound how well our algorithms will do in the worst case. A **minimax optimal** policy is a policy π which minimizes the supremum in the definition of minimax regret. That is, a minimax policy satisfies $R_n^*(\mathcal{E}) \geqslant R_n(\pi, E)$ for all $E \in \mathcal{E}$. It is almost impossible to find a minimax optimal policy, and it is not even guaranteed that such a policy exists. However, we shall often find **near minimax policies**, which are within a fixed, constant factor of $R_n^*(\mathcal{E})$, for all choices of n. If this is too much, we want log near minimax policies, which are bounding with a logarithmic factor of $R_n^*(\mathcal{E})$. Once we have found the appropriate lower bound for R_n^* over the class $\mathcal{E}_1$ of all 1-subgaussian bandits, we will have essentially verified that the upper confidence bound algorithm is optimal for the class of 1-subgaussian environments. The UCB gives us the bound

$$R_n^*(\mathcal{E}_1) \leqslant C\sqrt{Kn\log n}$$

We verify that this is with a log factor of the true optimum, so UCB is log near optimum of the set of 1 subgaussian environments. Thus we have set out to prove.

Theorem 2.1. *For any action space A, let $\mathcal{E}$ be the space of 1-subgaussian environments. Then there is a universal constant C such that*

$$R_n^*(\mathcal{E}) \geqslant C\sqrt{|A|n}$$

So that the bandit problem on 1-subgaussian is $\sqrt{n}$ hold.

2.1 Worst Case Lower Bounds

How on earth can we lower bound regret? The idea descends from information theory, and essentially the only way we can obtain lower bounds for statistical measurements. Consider the most basic problem, where we observe estimates $X_1,\dots,X_n$ from a gaussian distribution with unknown mean μ and variance $1/n$. We know either $\mu = 0$, or $\mu = \Delta$ for some $\Delta \in \mathbf{R}$, and we must decide which is true. If n is large compared to μ, then we should expect the problem to be easy, whereas if n is small the problem should be hard. Indeed, the concentration bound gives upper bounds

$$(2\pi n(\Delta/2)^2)^{-1/2}\exp(-n(\Delta/2)^2/2) \leqslant \exp(-\frac{n(\Delta/2)^2}{2})$$

so we have exponential decay in n. Using integration by parts, we can calculate that if $X \sim N(0,\sigma^2)$, then

$$\mathbf{P}(X > \varepsilon) \geqslant \left(\frac{\sigma}{\varepsilon} - \frac{\sigma^3}{\varepsilon^3}\right)\frac{\exp(-\varepsilon^2/(2\sigma^2))}{\sqrt{2\pi}}$$

This shows that any decision procedure with $n \leqslant 8c/\Delta^2$ will make a mistake with probability at least $Cc^{-1/2}(1-1/c)\exp(-c)$. The high level idea is to construct two environments which are impossible to separate with the number of samples given. We will perform the same technique on bandits, constructing bandits similar enough that cannot be distinguished, but such that a policy doing well on one instance cannot do well on the other instance, so an algorithm is forced to make a decision between the two instances. Thus we must reduce the bandit problem to a hypothesis testing problem. The lower bound has little interest in algorithms ran on a fixed instance, since we can often obtain much better bounds for specific instances (indeed, any environment has a policy that has zero, regret – the challenge is finding a policy that works uniformly well across all environments).

To prove this, we apply the High probability Pinsker bound of information theory, using the Kullback Leibler distance to measure the ability for a policy to distinguish between two environments. That is,

Theorem 2.2 (High Probability Pinsker). *Let μ and ν be probability measures on the same measure space. Then for any event A,*

$$\mu(A) + \nu(A^c) \geqslant \frac{1}{2}\exp(-D(\mu,\nu))$$

Where $D(\mu,\nu)$ is the Kullback Leibler distance betwee μ and ν.

Intuitively, if μ and ν are difficult to distinguish between, then we should expect $\mu(A) \approx \nu(A)$, and $\mu(A^c) \approx \nu(A^c)$, so $\mu(A) + \nu(A^c)$ should be large. For instance, for any δ, to guarantee that $\max(\mu(A), \nu(A^c)) \leqslant 1$, the Pinsker inequality tells us that we should have $D(\mu,\nu) \geqslant \log(1/4\delta)$. The theorem is useful for proving lower bounds, because it implies that $\mu(A)$ or $\nu(A^c)$ is large. If we make μ and ν be such that A is a 'bad' event for μ (a misclassification of a hypothesis), and A^c a 'bad' event for ν, then we can guarantee that any decision making procedure working of μ and ν could make a mistake with high probability.

So how do we formulate the problem with bandit optimality as a hypothesis testing problem? Given an environment μ, policy ν, and fixed horizon n, we let $\mathbf{P}_{\mu,\pi}$ be the distribution over histories of length n induced by the environment μ and policy π, and $\mathbf{E}_{\mu,\pi}$ the expectation with respect to this distribution. Our lower bound will depend on fixing π, and altering ν slightly so that π fails to realize the needed bound. First, note that

$$d\mathbf{P}_{\mu,\pi}(a_1, x_1, \dots, a_n, x_n) = \prod_{m=1}^{n} \pi(a_m | a_1, x_1, \dots, a_{m-1}, x_{m-1}) f^{\mu}_{a_m}(x_m | a_m) d\lambda(x_m) d\nu(a_m)$$

where λ is a dominating measure for the ρ_a, and $d\rho_a = f_a^\mu d\lambda$, and ν is the counting measure.

Lemma 2.3. *For any policy π,*

$$D(\mathbf{P}_{\pi,\mu}, \mathbf{P}_{\pi,\nu}) = \sum_a \mathbf{E}_{\pi,\mu}[T_a(n)] D(\mu(a), \nu(a))$$

Proof. First assume $D(\mu_a, \nu_a) < \infty$ for all a. Then by the chain rule for Radon Nikodym derivatives, and by using the decomposition formula above, the policy ratios cancel out and we find

$$\begin{aligned}
D(\mathbf{P}_{\pi,\mu}, \mathbf{P}_{\pi,\nu}) &= \mathbf{E}_{\pi,\nu}\left[\log\left(\frac{d\mathbf{P}_{\pi,\mu}}{d\mathbf{P}_{\pi,\nu}}\right)\right] \\
&= \sum_{a_1,\dots,a_n} \int \log\left(\frac{f_{a_1}^\mu(x_1)\dots f_{a_n}^\mu(x_n)}{f_{a_1}^\nu(x_1)\dots f_{a_n}^\nu(x_n)}\right) \\
&\quad \left(\prod_{m=1}^n \pi(a_m | a_1, \dots, x_{n-1}) f_{a_m}^\mu(x_m | a_m)\right) d\lambda(x_1)\dots d\lambda(x_n) \\
&= \sum_{m=1}^n \mathbf{E}_{\mu,\pi}[D(\mu_{A_m}, \nu_{A_m})] \\
&= \sum_a \mathbf{E}_{\mu,\pi}\left[\sum_{m=1}^n \mathbf{I}(A_m = a) D(\mu_a, \nu_a)\right] \\
&= \sum_a \mathbf{E}_{\mu,\pi}[T_n(a)] D(\mu_a, \nu_a)
\end{aligned}$$

And this was the formula we wanted to prove. □

Denote by $G_\mu \in \mathcal{E}_1$ the bandit environment where all distributions are Gaussian with unit variance and means $\mu \in [0,1]^A$. We let $R_n(\pi, E)$ denote the expected regret for a policy π on an environment E.

Theorem 2.4. *For any policy π, there is a mean vector $\mu \in [0,1]^A$ such that*

$$R_n(\pi, G_\mu) \geqslant \frac{1}{27}\sqrt{(|A| - 1)n}$$

Proof. Let $\Delta \in [0, 1/2]$ be the action gap, to be chosen later to optimize a bound. Pick some $a^* \in A$, and define

$$\mu(a) = \begin{cases} \Delta & a = a^* \\ 0 & \text{otherwise} \end{cases}$$

Now let $a' = \operatorname{argmin}_{b \neq a^*} \mathbf{E}_{\mu,\pi}[T_b(n)]$. Define a second environment

$$\mu'(a) = \begin{cases} \Delta & a = a^* \\ 2\Delta & a = a' \\ 0 & \text{otherwise} \end{cases}$$

Note that the optimal action in the first action is a^*, whereas in the second algorithm the best action is a'. Now if π chooses action a^* more often, it will likely succeed on G_μ, but not $G_{\mu'}$. Conversely, if π chooses a', then it will succeed on $G_{\mu'}$, but not G_μ. In particular

$$R_n(\pi, G_\mu) \geqslant \mathbf{P}_{\mu,\pi}(T_n(a^*) \leqslant n/2)\frac{n\Delta}{2} \qquad R_n(\pi, G_{\mu'}) \geqslant \mathbf{P}_{\mu',\pi}(T_n(a^*) > n/2)\frac{n\Delta}{2}$$

The Pinsker inequality implies, since $\mathbf{E}_{\mu,\pi}[T_{a'}(n)] \leqslant n/(|A|-1)$, by the minimality choice,

$$\begin{aligned} R_n(\pi, G_\mu) + R_n(\pi, G_{\mu'}) &\geqslant \frac{n\Delta}{2}\Big(\mathbf{P}_{\mu,\pi}(T_n(a^*) \leqslant n/2) + \mathbf{P}_{\mu',\pi}(T_n(a^*) > n/2)\Big) \\ &\geqslant \frac{n\Delta}{4}\exp(-\sum_a \mathbf{E}_{\pi,\mu}[T_n(a)]D(N(\mu_a, 1), N(\mu'_a, 1)) \\ &= \frac{n\Delta}{4}\exp(-2\mathbf{E}_{\pi,\mu}[T_n(a')]\Delta^2) \\ &= \frac{n\Delta}{4}\exp(-2n\Delta^2/(|A|-1)) \end{aligned}$$

If we tune $\Delta^2 = (|A|-1)/4n$, we obtain the required bound. □

Now we assumed that the class of 1-subgaussian bandits obtained the Gaussian distributions, we only ever used the fact that the class contained probability distributions ρ^μ with certain means μ_a, such that $D(\rho_a^\mu, \rho_a^{\mu'}) = O((\mu_a - \mu'_a)^2)$. This is certainly not true of all families of distributions, but we should expect it to be true for a large class of algorithms, since if we have a parametric family of functions whose densities are twice differentiable, we can approximate these parameters by the second derivative, and this gives us bounds of the form needed above.

Note that UCB is log near optimal, but a slight modification of UCB, known as MOSS, can obtain near optimal estimates by chopping off the log factor. Unfortunately, we do not have time to discuss it here, so we refer to the paper by Bubeck and Audibert.

2.2 Instance Dependent Lower Bounds

Just because the UCB algorithm is log near optimal for the class of 1 subgaussian environments, is not a sufficient argument to justify that the algorithm is necessarily is the correct algorithm to use over all 1-subgaussian environments. Indeed, consider the modification of the UCB algorithm, which takes $0 < D \ll C$, where C is the constant in the upper bound for UCB. The modified version of UCB explores in the first $m = D\sqrt{|A|n}$ rounds, and then switch to the UCB policy. It is not difficult to see that this version of UCB is also log near optimal in the worst case.

So why do we feel like UCB is still a better algorithm? Consider a 1-subgaussian environment ν where choosing any action suboptimally gives you a reward close to one. Then UCB will stop using suboptimal actions very quickly, so that UCB would achieve small regret. Indeed, the upper bound gives

$$R_n \leqslant \sum_a \frac{C\log(n)}{\Delta_a} \sim C|A|\log n$$

whereas the new exploratory version of UCB we defined obtains $C'\sqrt{|A|n/2}$ guaranteed regret, which for large n will eventually be larger than the other bound. The problem is that even though UCB and exploratory UCB obtain essentially the same regret in the worst case, UCB tends to achieve much better regret bounds on easier instances of the problem.

Thus we desire our algorithms to not only be bad on worst-case instances, but also get better regret bounds on 'nice' instances on the problem. We have shown that in certain instances UCB can obtain logarithmic regret. One difficulty of a particular instance of the bandit problem could be measured as

$$\limsup_{n\to\infty} \frac{R_n}{\log n}$$

We shall show that in some sense, the logarithmic regret bound for UCB on certain problems is the best possible. No *reasonable* strategy can beat UCB on any instance of the bandit problem. This is an important result, separate from the worst case regret bounds, for it shows why UCB can be successfully applied to both reasonable and unreasonable environments, with good asymptotic regrets in both, relative to the environment chosen. However, the logarithmic measure of difficulty is perhaps overly restrictive, so we will only require that the policy chosen has subpolynomial growth.

Let us rigorously define these ideas. A policy π over $\mathcal{E}$ is **consistant** if for any environment $\mu \in \mathcal{E}$, and $p > 0$,

$$R_n(\pi, \mu) = O_{p,\mu}(n^p)$$

The set of consistant policies is denoted $\Pi_{\text{cons}}(\mathcal{E})$. The UCB policy is consistant, because it gives asymptotically logarithmic regret over individual environments. The main theorem of this section is that UCB is the best consistant policy over Gaussian bandits. Let $\mathcal{E}_N$ be the class of bandits whose arms have unit variance normal distributions.

Theorem 2.5. *For any consistant policy π over $\mathcal{E}_N$, and $\nu \in \mathcal{E}_N$,*

$$\liminf_{n \to \infty} \frac{R_n(\pi \nu)}{\log n} \geqslant \sum_a \frac{2}{\Delta_a(\nu)}$$

so that UCB is optimal to a fixed constant factor.

Proof. We apply the Pinsker inequality over Gaussian environments. Consider a sequence of means μ, pick some action a', and consider a new sequence

$$\mu'(a) = \begin{cases} \mu(a) + \lambda & a = a' \\ \mu(a) & \text{otherwise} \end{cases}$$

Let $A = \{T_{a'}(n) \geqslant n/2\}$, so $A^c = \{T_{a'}(n) < n/2\}$. We have

$$R_n(\mu, \pi) \geqslant \frac{n\Delta_{a'}}{2} \mathbf{P}_{\mu,\pi}(A) \qquad R_n(\mu', \pi) \geqslant \frac{n(\lambda - \Delta_{a'})}{2} \mathbf{P}_{\mu',\pi}(A^c)$$

□

Chapter 3

Adversarial Bandits

In many situations, we want to treat a problem like a bandit problem, even though there are no guarantees of stationarity in the distribution. For instance, we may be trading stocks, in which case the action of buying a stock is certainly non stationary over time. We also should avoid using the upper confidence bound algorithm, because other traders may catch onto our strategy and exploit our optimism for their own ends. The adversarial environment provides theoretical guarantees on the success of these algorithms, by assuming there's a person on the other end of the algorithm setting the rewards to screw you over. This implies that regardless of the rewards we actually get, we will do better than if there was someone purposefully trying to cause you to get the worst reward.

The reward distribution of the adversarial bandit problem is completely non-stationary. We consider an environment $\nu \in [0,1]^{n \times A}$ (we have to assume the rewards are bounded, or else the adversarial problem cannot really be specified – someone can always cause you to lose an unbounded amount of money). A policy is specified as in the stochastic bandit problem, giving a probability distribution over histories, and determining a sequence of actions and rewards

$$A_1, X_1, A_2, X_2, \dots$$

where $(X_i | A_i = a) \sim \nu(i,a)$ (reward is deterministic, though the choice of action is not). The adversarial regret of a policy π with respect to the environment ν is

$$R_n(\pi, \nu) = \mathbf{E}\left[\left(\max_a \sum_{i=1}^n \nu(i,a) \right) - \sum_{i=1}^n X_i \right]$$

the adversarial regret measures the difference between the optimal 'expoit' strategy, and your policy. Given a fixed ν, it is always possible to obtain zero regret, and there is even the possibility of negative regret (how?)!

The difference between the adversarial bandit problem and the stochastic bandit problem is that the goal is not to optimize over a specific environment. Instead, we aim to find a policy which obtains a best worst case regret over all policies. That is, we wish to minimize the minimax regret

$$R_n^*(\pi) = \sup_{\nu \in [0,1]^{n \times A}} R_n(\pi, \nu)$$

Game theoretically, we choose a policy, and then our opponent gets to pick the worst environment for our policy. The main question is whether we can find policies for which $R_n^*(\pi)$ is sublinear (as $n \to \infty$).

Unlike in the stochastic problem, deterministic policies are completely unusable in the adversarial setting. If an opponent knows your strategy for a game, then it is very easy to take advantage of your choices. Consider the actions $a_1, a_2, \dots, a_n$ for the deterministic policy, assuming that $\nu(i, a_i) = -1$ for all i, and then assign $\nu(j, a_j) = 1$ for all other actions. Then we have

$$R_n(\pi, \nu) = n + \max \left(\sum_{i=1}^n \nu(i, a) \right) \geqslant n + n(1 - 2/|A|)$$

Which gives us at least linear regret for $R_n^*(\pi)$. This tells us that good algorithms for the adversarial problem should be random, and not just random, but unpredictably random.

Note, however, that an adversarial policy will perform at least as good in the stochastic setting as it does in the adversarial setting, for the stochastic regret will always be pointwise better than the minimax regret. This means we may apply the lower bounds we have obtained before to conclude that $R_n^* \geqslant C\sqrt{n|A|}$ for some constant C. We cannot directly apply the lower bound for bandits, since there we had Gaussian payoffs, but we can provide a very similar argument for Bernoulli bandits, which do apply to this situation.

The algorithm that is near minimax adversarial optimal is the EXP3 algorithm (Exponential weight algorithm for Exploration and Exploitation). The idea is that, since rewards are bounded, there is no way to 'sneak' a best policy past an agent. In order for an action to be the 'best', it must consistently give good rewards, and we can take advantage of that.

In order to estimate the best action, we keep track of a probability distribution over actions, which we use to sample the next action, and the obtained reward will be used to update the distribution. Initially, the distribution is uniform. Consider the **importance sampling estimators**

$$\widehat{X}_{n,a} = \frac{\mathbf{I}(A_n = a)}{\mathbf{P}(A_n = a | H_{n-1})} X_n$$

The estimates are random, and

$$\mathbf{E}[\widehat{X}_{n,a} | H_{n-1}] = \mathbf{E}(X_n | H_{n-1}, A_n = a) = \nu(n,a)$$

and they are therefore unbiased estimates. We similarily calculate the second moment

$$\mathbf{E}(\widehat{X}_{n,a}^2 | H_{n-1}) = \frac{\nu(n,a)^2}{\mathbf{P}(A_n = a | H_{n-1})}$$

and therefore the variance

$$\mathbf{V}(\widehat{X}_{n,a} | H_{n-1}) = \frac{1 - \mathbf{P}(A_n = a | H_{n-1})}{\mathbf{P}(A_n = a | H_{n-1})} \nu(n,a)^2$$

This variance explodes as the probability of choosing a particular action decreases to zero. A similar estimator is

$$\widehat{X}_{n,a} = 1 - \frac{\mathbf{I}(A_n = a)}{\mathbf{P}(A_n = a | H_{n-1})}(1 - X_n)$$

The estimator is unbiased, and if we set $Y_n = 1 - X_n$, and $\widehat{Y}_{n,a} = 1 - \widehat{X}_{n,a}$, then the above formula can be reexpressed as

$$\widehat{Y}_{n,a} = \frac{\mathbf{I}(A_n = a)}{\mathbf{P}(A_n = a | H_{n-1})} Y_n$$

So the estimate is essentially the same old importance sampling estimator, but for Y_n. We call the Y_n losses, and the estimator above the loss based importance sampling estimator. This is the first time we have met losses, but we could have actually defined the whole bandit regime in terms of losses, where we attempt to minimize loss instead of maximizing reward (except this would only really work on a bandit problem). The loss based importance sampling estimator will give better results if the losses are on average smaller than the rewards. Note that while the two estimators have the same mean, the first estimator takes values in $[0,\infty)$ and the second estimator takes values in $(-\infty,1]$.

With the estimates of the future actions in hand (where we can, on average, somehow manage to estimate rewards we haven't seen yet), we can define the estimated total reward at the end of round n for action a as

$$\widehat{S}_{n,a} = \sum_{m=1}^{n} \widehat{X}_{m,a}$$

We then use these estimated totals to perform a weighted average over the next action to select. The standard average to take is an exponential average, taking the probability distribution

$$\mathbf{P}(A_n = a|H_{n-1}) = \frac{e^{\eta \widehat{S}_{n-1,a}}}{\sum e^{\eta \widehat{S}_{n-1,b}}}$$

where $\eta > 0$ is a tuning parameter which controls how fast we change probabilities over time. As $\eta \to \infty$ the policy becomes more 'greedy'.

As with the mean sampling estimates, it is useful to note that

$$\mathbf{P}(A_{n+1} = a|H_n) = \frac{\mathbf{P}(A_n = a|H_{n-1}) e^{\eta \widehat{X}_{n,a}}}{\sum_b \mathbf{P}(A_n = b|H_{n-1}) e^{\eta \widehat{X}_{n,b}}}$$

The algorithm is tricky to implement while also being numerically stable for a large number of rounds. One trick is to calculate $\tilde{S}_{n,a} = \widehat{S}_{n,a} - \min_b \widehat{S}_{n,b}$, and then write

$$\mathbf{P}(A_{n+1} = a|H_n) = \frac{e^{\eta \tilde{S}_{n,a}}}{\sum_j e^{\eta \tilde{S}_{n,b}}}$$

This saves us from taking the exponential of too large a value.

Let us prove upper bounds on the minimax regret of this policy.

Theorem 3.1. *For any environment $\nu \in [0,1]^{n \times A}$, the regret of the EXP3 algorithm satisfies*

$$R_n \leqslant \sqrt{2n|A|\log|A|}$$

Proof. We can reduce the theorem to bounding

$$R_{n,a} = \sum_{m=1}^{n} \nu(m,a) - \mathbf{E}\left[\sum_{m=1}^{n} X_m\right]$$

Note that $\mathbf{E}(\widehat{S}_{n,a}) = \sum_m \nu(m,a)$, since the $\widehat{X}_{n,a}$ are unbiased, and

$$\mathbf{E}(X_n|H_{n-1}) = \sum_a \mathbf{P}(A_n = a|H_{n-1})\nu(n,a) = \sum_a \mathbf{P}(A_n = a|H_{n-1})\mathbf{E}(\widehat{X}_{n,a}|H_{n-1})$$

It follows that

$$\mathbf{E}\left(\sum_{k=1}^{n} X_k\right) = \mathbf{E}\left(\sum_{k=1}^{n}\sum_a \mathbf{P}(A_n = a)\widehat{X}_{n,a}\right)$$

If we define

$$\widehat{S}_n = \sum_a \sum_{m=1}^{n} \mathbf{P}(A_m = a|H_{m-1})\widehat{X}_{m,a}$$

It suffices to bound the expectation of $\widehat{S}_{n,a} - \widehat{S}_n$, and we do this by taking exponentials. Denote by $W_n = \sum_a e^{\eta \widehat{S}_{n,a}}$. We let $\widehat{S}_{0,a} = 0$, so that $W_0 = |A|$ and so for any b

$$e^{\eta \widehat{S}_{n,b}} \leqslant \sum_a e^{\eta \widehat{S}_{n,a}} = W_n = W_0 \frac{W_1}{W_0}\frac{W_2}{W_1}\cdots\frac{W_n}{W_{n-1}}$$

and

$$\frac{W_k}{W_{k-1}} = \sum_a \frac{e^{\eta \widehat{S}_{k-1,a}}}{W_{k-1}} e^{\eta \widehat{X}_{k,a}} = \sum_a \mathbf{P}(A_k = a|H_{k-1})e^{\eta \widehat{X}_{k,a}}$$

Since $\widehat{X}_{k,a} \leqslant 1$ (we're using the loss based estimate), and for $x \leqslant 0$, $e^x \leqslant 1 + x + x^2/2$, thus

$$e^{\eta \widehat{X}_{k,a}} = e^{\eta} e^{\eta(\widehat{X}_{k,a}-1)} \leqslant e^{\eta}\left[1 + \eta(\widehat{X}_{k,a} - 1) + (\eta^2/2)(\widehat{X}_{k,a} - 1)^2\right]$$

and so

$$\begin{aligned}\frac{W_k}{W_{k-1}} &\leqslant \sum_a \mathbf{P}(A_k = a|H_{k-1})e^{\eta \widehat{X}_{k,a}} \\ &\leqslant \sum_a e^{\eta}\mathbf{P}(A_k = a|H_{k-1})\left[1 + \eta(\widehat{X}_{k,a} - 1) + (\eta^2/2)(\widehat{X}_{k,a} - 1)^2\right] \\ &= (1 - \eta + \eta^2/2)e^{\eta} + \eta e^{\eta}(1-\eta)\sum_a \mathbf{P}(A_k = a|H_{k-1})\widehat{X}_{k,a} \\ &\quad + e^{\eta}(\eta^2/2)\sum_a \mathbf{P}(A_k = a|H_{k-1})\widehat{X}_{k,a}^2 \\ &\leqslant e^{\eta \sum_a \mathbf{P}(A_k=a|H_{k-1})\widehat{X}_{k,a}} e^{(\eta^2/2)\sum_a \mathbf{P}(A_k=a|H_{k-1})(\widehat{X}_{k,a}-1)^2}\end{aligned}$$

Now we put these two inequalities together to find

$$e^{\eta \widehat{S}_{n,b}} \leqslant |A| e^{\eta \sum_{k=1}^n \sum_a \mathbf{P}(A_k=a|H_{k-1})\widehat{X}_{k,a}} e^{(\eta^2/2)\sum_{k=1}^n \sum_a \mathbf{P}(A_k=a|H_{k-1})(\widehat{X}_{k,a}-1)^2}$$

Hence

$$\widehat{S}_{n,b} - \widehat{S}_n \leqslant \frac{\log|A|}{\eta} + (\eta/2)\sum_{k=1}^n \sum_a \mathbf{P}(A_k = a|H_{k-1})\widehat{Y}_{k,a}^2$$

Now it just remains to bound $\sum_{k=1}^n \sum_a \mathbf{P}(A_k = a|H_{k-1})\widehat{Y}_{k,a}^2$, and since $\mathbf{P}(A_k = a|H_{k-1})\widehat{Y}_{k,a} \leqslant 1$, we find that the total is bounded by $n|A|$ and so

$$\widehat{S}_{n,b} - \widehat{S}_n \leqslant \frac{\log|A|}{\eta} + (\eta n|A|/2)$$

And choosing η to minimize the inequality gives us the needed result. □

Thus we obtain good expected results. But

$$\widehat{R}_n = \max_a \sum_{m=1}^n \nu(m,a) - X_m$$

is not necessarily small. We cannot guarantee its small, but we should be able to get a high probability bound on such an occurence not happening. Since the rewards in the adversarial case are not independant, the rewards are highly correlated, which means that we cannot get a high probability bound with the vanilla EXP3 algorithm. This occurs because the tail bound is related to the variances of the regret, which normally break up linearly when the rewards are independant.

It shall be convinient now to switch to losses, and write

$$\widehat{L}_n - \widehat{L}_{n,a} \leqslant \frac{\log |A|}{\eta} + \frac{\eta}{2} \sum_a \widehat{L}_{n,a}$$

where

$$\widehat{L}_n = \sum_{k=1}^n \sum_a \mathbf{P}(A_k = a | H_{k-1}) \widehat{Y}_{k,a} \qquad \widehat{L}_{n,a} = \sum_{k=1}^n \widehat{Y}_{k,a}$$

Suppose we also define

$$\tilde{L}_n = \sum_{m=1}^n Y_t \quad L_{n,a} = \sum_{m=1}^n (1 - \nu(m,a))$$

The random regret is the max of $\tilde{L}_n - L_{n,a}$ over all choices of a, and thus it just remains to bound this value.

To bounds these values, we need to slightly modify the reward estimates. A simple fix to avoid estimates shooting up is to fix $\gamma > 0$ and define

$$\widehat{X}_{n,a} = 1 - \frac{\mathbf{I}(A_n = a)}{\mathbf{P}(A_n = a | H_{n-1}) + \gamma}(1 - X_t)$$

This modified version of the EXP3 algorithm is called the EXP3-IX algorithm, since the algorithm now 'xplores implicitly'.

The γ certainly biases the estimator, and the bias is upward, in that $\widehat{X}_{n,a}$ is more optimistic.

3.1 Contextual Bandits

We now introduce some state to the bandit problem. In the contextual bandit problem, in each round we recieve a context c from a finite set of contexts C, we choose some action $a \in A$ and we recieve a reward sampled from $r(c,a) + \eta$, where η is σ^2-subguassian, given past knowledge. The aim is to minimize the regret

$$R_n = \mathbf{E}\left[\sum_{m=1}^n \max_c r(c,a) - \sum_{m=1}^n X_m \right]$$

In the linear contextual bandit problem, we are given a function $\varphi : C \times A \to \mathbf{R}^d$, and we assume there is $\theta \in (\mathbf{R}^n)^*$ such that $r(c,a) = \theta(\varphi(c,a))$. This makes the problem much more feasible.

3.2 Nonstationarity

A final problem we will discuss with bandit problems is the problem of nonstationarity. Regardless of how long a policy interacts with a bandit, the reward distributions R_a remain the same. However, what if these values do change according to some rule, that is, if there is instead a sequence of functions $(R_a^1, R_a^2, \dots)$, which we sample from in each interval. Provided these functions have some relation between them, there is still a chance a policy could find a good strategy for obtaining award by carrying experience between choices. But now regret is not well defined, as μ_* changes over time. It follows that computing the estimates $\widehat{\mu_n}$ will no longer lead to good performance. Note that the recurrence relation

$$\widehat{\mu_a}(n+1) = \widehat{\mu_n}(a) + \frac{\mathbf{I}(A_{n+1} = a)}{T_a(n+1)}[X_{n+1} - \widehat{\mu_n}(a)]$$

It is of the form

$$\text{‘new average’} = \text{‘old average’} + \text{‘step size’}[\text{‘new value’} - \text{‘old average’}]$$

If we adjust the step size from $T_a(n+1)$ to a non-zeroconstant value α, our sample average will not settle down to some value, but will still have some convergence to the mean, so that we have a new estimate

$$\begin{aligned}\widehat{\mu_a}(n+1) &= \widehat{\mu_a}(n) + \alpha \mathbf{I}(A_{n+1} = a)[X_{n+1} - \widehat{\mu_a}(n)] \\ &= (1-\alpha)^{T_a(n+1)}\widehat{\mu_a}(a) + \alpha \sum_{k=1}^{n} \mathbf{I}(A_k = a)(1-\alpha)^{T_a(n+1) - T_a(k)} X_k\end{aligned}$$

We call this the exponential recency weighted average, as past results decrease exponentially in importance as new results are obtained. This allows us to get some convergence in the mean, while still not settling down completely. We can even change the step size α for each update to some sequence α_k. If the mean of the rewards is fixed, $\sum \alpha_k = \infty$ and $\sum \alpha_k^2 < \infty$, then the law of large numbers continues to hold for our estimates. Alas, then $\alpha_k \to 0$, so we run into the same problem as the standard sample averages.

3.3 Confidence Intervals for Linear Bandits

s

3.4 Adversarial Linear Bandits

We can generalize EXP to linear bandits, when we look at linear bandits that are adversarial. We consider loss vectors $y_1, y_2, \cdots \in \mathbf{R}^n$, and we assume that each action is identified with a vector $a \in \mathbf{R}^n$. In the one dimensional adversarial problem, $-1 \leqslant 0 \leqslant 1$. In this version of adversarial bandits, we assume $|\langle y_1, a\rangle| \leqslant 1$. We obtain loss estimates $\hat{Y}_n(a)$, and define probabilities

Chapter 4

Partial Monitoring

The partial monitoring problem models the bandit problem with the additional difficulty that our observation of rewards is obscured. For example, consider $W_1, W_2, \dots \sim \text{Ber}(p)$, where $p \in [0,1]$ is unknown. In each round, we have 3 actions A, B, C, and

$$(X_k | A_k = A) \sim W_k \quad (X_k | A_k = B) \sim 1 - W_t \quad (X_k | A_k = C) \sim -1$$

So that in the normal bandit problem, choosing C would always be an accident. But suppose that we only observe W_t when we select action C. Then the problem becomes more interesting, for we must choose action C to be able to statistically determine whether A or B is more optimal.

Consider an explore and commit approach, where we attempt to estimate p until we are confident enough that one action is more optimal. Since W_t is one-subgaussian, if we sample W_t n times than

$$\mathbf{P}(|p - \widehat{p}| < x) = 1 - \mathbf{P}(|p - \widehat{p})| \geqslant x) \geqslant 1 - 2e^{-(xn)^2}$$

So a confidence interval for $1 - \delta$ is $x = \sqrt{\log(2/\delta)}/n$. If we can conclude that $\widehat{p} > 1/2$ with some confidence, it may be wise to play action B. If $\widehat{p} < 1/2$, we should play action A. The only problem is being able to conclude that $\widehat{p} = 1/2$, which we can only do with some confidence.

We shall consider the adversarial partial monitoring problem. We have finitely many actions A, states E, and observations O we have a loss function $L(a,e) \in \mathbf{R}$, and an observed function $\Sigma(a,e) \in O$. The regret is then defined as

$$R_n = \sum_{m=1}^n L(a_m, e_m) - \min_a \sum_{m=1}^n L(a, e_m)$$

The L and Σ are given to you to optimize over, but the adversary can choose e_m at whim. The proper way to think about Σ is a way to count frequencies of observations.

We can reformulate the problem as being very nearly a linear bandit problem. Given losses $l_a \in [0,1]^E$, and actions $S_a : E \to O$. The states of the problem are chosen by the opponent $e_1, \dots, e_n \in E$. The player then chooses an action A_k on round k, observes $S_{A_k}(e_k)$. The reward is

$$R_n = \mathbf{E}\left[\sum l_{A_k}(e_m) - \min_{a \in A} \sum_{m=1}^{n} l_a(e_m)\right]$$

Consider the averages

$$q_n = \frac{1}{n} \sum_{m=1}^{n} e_m \in \mathbf{R}\langle E \rangle$$

which move around the simplex Δ_E of points p with $\sum_e p(e) = 1$, which we can interpret as the space of probability distributions over possible events. We can see the adversaries problem as manipulating the motion of the sequence $q_1, q_1, \dots, q_n$ in such a way that $l_a(q_n)$ is maximized, in such a way that is confuses the player as much as possible.

We can decompose the simplex Δ_E into cells which correspond to actions. Indeed, we let C_a be the set of distributions p such that a minimizes $l_a(p)$. These are cells, because they are the intersection of half planes – the half planes defined by $l_a(p) \leqslant l_b(p)$, for other actions b. We say a and b are neighbors if $C_a \cap C_b \neq \varnothing$. We say a and b are BC if there are $p, q \in \Delta_E$ for which $S_a p = S_a q$, $S_b p = S_b q$, and $\operatorname{sgn}((l_a - l_b)(p)) \neq \operatorname{sgn}((l_a - l_b)(q))$. There is an effective way to determine if two actions are BC – $l_a - l_b \notin \operatorname{Im}(S_{ab}^t)$, where $S_{ab}(p) = (S_a(p), S_b(p))$. This is just checked by linear algebra.

Now suppose p_0 is an interior point of Δ_E, with $q = p_0 + \varepsilon v \in C_a$, $q' = p - \varepsilon v \in C_b$. We normalize v such that $(l_a - l_b)(q) = \varepsilon$, so $(l_b - l_a)(q') = \varepsilon$. Suppose that the adversary draws events $Y_1, Y_2, \dots, Y_n$ randomly according to the distribution q, and $Y_1', \dots, Y_n'$ from q'. Then

$$R_n = \mathbf{E}\left[\sum_{m=1}^{n} l_{A_k}(Y_k) - \min_a \sum_{m=1}^{n} l_a(Y_k)\right]$$

$$R_n' = \mathbf{E}\left[\sum_{m=1}^{n} l_{A_k}(Y_k') - \min_a \sum_{m=1}^{n} l_a(Y_k')\right]$$

If we let T_a T_b count the number of times we chose action b, we can spit the reward up and apply the Pinsker bound to obtain $\Omega(n^{2/3})$ lower bounds on the regret of any policy in partial monitoring.

The good case of partial monitoring occurs when all actions are distinguishable. We expect good results here, because adversarial bandits lie here (where we see the state as our observation). Suppose there are no dominated actions (for which $C_a = \varnothing$), duplicated actions ($l_a = l_b$), or degenerate actions (there is b with C_a a proper subset of C_b). Then the minimax regret satisfies $R_n^*(G) = \inf_\pi \sup_{E \in \mathcal{E}} R_n^\pi(E, G)$. Then either $R_n^*(G) = 0$, if there is one action, $\Theta(n^{1/2})$, if there is more than one action and G is locally observable, or $\Theta(n^{2/3})$ if G is globally observable but not locally observable (locally observable means any two actions can be distinguished by their losses, globally observable means $l_i - l_j \in \oplus_a \text{Im}(S_a^t)$). If G is not globablly observable, the reward is $\Theta(n)$. (The Θ's may obscure logarithm factors).

To find a good algorithm on a **point-local process**, where the intersection of all C_a is nosn-trivial, is to take estimates $\hat{l}_a$ of the losses, and define $\hat{P}_{n,a} = (1-\gamma)P_{n,a} + \gamma \frac{1}{|A|}$, where

$$P_{n,a} = \frac{\exp(-\eta \sum_{m=1}^{n-1} \hat{l}_{m,a})}{\sum_b \exp(-\eta \sum_{m=1}^{n-1} \hat{l}_{m,b})}$$

Where

$$\hat{l}_{n,t} = \left(\frac{A_{n,a}}{\hat{P}_{n,a}} v_{a,A'_n} - v_{A_n,a} \right)^T Y_n$$

and A_n, A'_n are two actions drawn from $\hat{P}_n$. Then $\hat{l}_{n,b} - \hat{l}_{n,b}$ is an unbiased estimator of $(l_a - l_b)(y_n)$. The standard proof for exponential weighting owrks.

Chapter 5

Markov Decision Processes

The next step to adding to the learning process is the introduction of state, the 'colours' of the bandit problem discussed at the end of the last chapter represent what we mean by the word state, a summary of experiences relevant enough to predict the present to the best of our abilities. Actions thus have further consequences than immediate reward. We call this associate learning problem the reinforcement learning problem.

The reinforcement learning task consists of an agent/learner/decision-maker interacting with its environment, everything that is outside of the agents direct control. The two interact continually through the environments presentation of rewards to the agents actions in the environment, providing 'reinforcement' that allows the agent to improve its behaviour over time, the agents primary motive. Over time, the environment changes, but the environment should give enough information at the current time or via previous interactions to allow an agent to form a state representation of the agents current standing with the markov property - the presense of enough relavant information to make intelligent decisions based on previous information about the same state. We define the reinforcement learning task rigorously through the introduction of a markov decision problem, or MDP.

A Markov Decision Problem, or MDP, is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P} \rangle$, where S is a non-empty set of states, A is a non-empty set of actions, R is a mapping from an initial state, an action, and a resultant state, $S \times A \times S$, to a distribution over the real numbers called the reward distribution (think of the reward distribution from the bandit problem), and P is a mapping from $S \times A$ to a probability distribution over states, called the transition probability kernel. We write

$$R(\cdot|s,a) \qquad\qquad P(\cdot|s,a)$$

for the reward and probability distributions mapped from state s and action a. Each state s has associated with it a subset of A called the admissible actions at state s. We write this as $\mathcal{A}_s$. The set of states, actions, and rewards are assumed to be countable in this course, though it should be said that this need not always be assumed. An experience in a markov decision process is a sequence with elements in S, A, and R, in that order, normally denoted

$$(S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_n, A_n, R_{n+1})$$

We write R_{t+1} instead of R_t in the sequence to represent the fact that R_t is created leading into the next time period. A policy π is then a mapping from experience to a probability distribution over the admissible actions in the most recent state seen. The aim of a policy is to maximise its return, the expectation of reward the policy may see. We denote the expected reward after t steps by $\mathbf{E}_\pi[G_t]$. We define the expectation by the following equations based on the initial state S_0:

$$\mathbf{E}_\pi[G_t] = \sum_{k=0}^{\infty} \gamma^k R_k$$

$$R_0(S_0, A_0, R_1, \ldots, S_n) = \sum_{a,s,r} \pi(a|S_0, A_0, \ldots, S_n) P(s|S_n, a) \mathcal{R}(r|S_n, a, s) r$$

$$R_t(S_0, \ldots, S_n) = \sum_{a,s,r} \pi(a|S_0, A_0, \ldots, S_n) P(s|S_n, a) R_{t-1}(S_0, \ldots, S_n, a, r, s)$$

Intuitively, given some initial state S_0, the policy picks an action, gets a reward from it, and moves to a new state. Given the new information, it picks an action, gets a reward, and the process continues. R_i then becomes the expected reward from the t'th action given we are using policy π. The optimal function π_* maximises this value.