Reinforcement Learning

Jacob Denson

November 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. We note that the problem is not just a toy problem for mathematicians. 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. 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 boon to those who need the solutions.

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 ρ_a over the real numbers. The set of all 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 choose a sequence of actions to maximize our reward, by learning which actions consistently gives us the highest reward, in the fewest exploratory moves possible (or at least, to explore

only on states that won't cause us to miss out on rewards). Technically, the domain we are defining is the stochastic, stationary bandit problem, but for now we shall just call the problem the bandit problem.

After a bandit problem has been specified, we must decide our strategy to maximize reward? The key definition is a policy 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 $\pi_i : \mathcal{H}_{i-1} \to A$, which decide based on prior experience what to take as an i'th action. The decision function induces a random sequence of actions $A_1, A_2, \ldots$, rewards $X_1, X_2, \ldots$, and histories $H_k = [(A_1, X_1), \ldots, (A_k, X_k)]$, 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** is given by $\pi_i : \mathcal{H}_{i-1} \to P(A)$, a sequence which now map into the space of all probability measures on A. A stochastic policy the induces a sequence $A_1, A_2, \ldots$ and rewards $X_1, X_2, \ldots$, together with a filtration $\{\mathcal{F}_1, \mathcal{F}_2, \ldots\}$ 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

$$\mathbf{E}\left[\sum_{k=1}^{n} X_k\right]$$

for some value $n \in \mathbf{N}$ (the 'finite horizon problem') or for $n = \infty$ (the infinite horizon problem), in which case we try to optimize the order of growth of the expected reward as $n \to \infty$.

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 most 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, which still optimizing the reward obtained. We see therefore that this problem possesses 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 to achieve maximal rewards.

1.1 The Regret Function

Another way to visualize the problem is that rather than maximizing 'good' decisions, but to 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, the regret is obtained by a shift and sign change. However, the regret is a much more meaningful quantity when we consider algorithms over certain classes of different environments, where the reward distributions may be unbounded, but the differences $(\mu^* - \mu_a)$ bounded.

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. Thus we keep the notation above for use later.

One problem with calculating the means above is that to calculate the means of each action we need to store the values $T_n(a)$ $\mathbf{I}(A_k = a)$, and X_k, and the we can calculate this in $\Theta(n^2|A|)$ with $\Theta(n|A|)$ memory usage. There is a computational trick for keeping track of these estimates without having to store the values of the indicator function for all k, reducing the computation time to amortized $\Theta(|A|)$ time. We have

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

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.

1.2 Naive, Epsilon-Greedy, and Softmax Policies

A naive policy, the greedy method, always takes the action with the highest sample average. The problem with this method is, if the reward for some action gets distorted low enough, 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 no guarentees are set. One other 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, there is not really a way of mathematically guaranteeing the success of this policy, since the effects of initial choices in the algorithm give rise to combinatorial number of different outcomes. A 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 such that $0 \leqslant \varepsilon \leqslant 1$. The idea of the ε greedy method is to act greedily most of the time, whereas for some proportion of the time related to ε, we act randomly, ensuring we sample enough time to get convergence of our sample averages. The ε greedy policy π is defined for each action a by the distribution

$$\mathbf{P}(\pi_k = a | H_{k-1}) = \begin{cases} \varepsilon + \frac{1-\varepsilon}{|A|} & : a = \operatorname{argmax}_{a' \in A} \widehat{\mu_{a'}}(k) \\ \frac{1-\varepsilon}{|\mathcal{A}|} & : \text{otherwise} \end{cases}$$

We can use the regret decomposition formula to conclude that

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

so that we have $\Theta(n)$ expected regret; not the best situation to be in, but a good start. With some work, and further specializing the distributions of the arms, we can show that $R_n \sim n\varepsilon(\sum \Delta_a)/|A|$.

While ε-greedy policies are a simple method that converge to a close to optimal policy, if there is one action a with a very low expected value, the regret for π will be strongly affected (in other words, if the average value of the action gap is large). Take for example the problem of helping a patient with the possibility of diabetes, where we can either not give

the patient insulin, give the patient insulin, or chop off the patient's legs. The softmax method attempts to fix this 'wanton exploration' by choosing actions randomly, just with a higher probability of taking actions with a higher average reward.We select an action more probably if it has a higher sample average. A softmax method with parameter τ is the policy defined by

$$\mathbf{P}(\pi_k = a | H_{k-1}) = \frac{e^{\widehat{\mu_a}(k)/\tau}}{\sum_{a'} e^{\widehat{\mu_{a'}}(k)/\tau}}$$

In the limit for nice distrubtions we should have

$$R_n \sim \sum_a \frac{e^{\mu_a/\tau}}{\sum_{a'} e^{\mu_{a'}/\tau}} \Delta_a$$

As τ decreases in value, the disparity in probabilities between actions increases, causing the policy to act more greedily. Besides the convergence of the policy, there are no mathematical guarentees on which τ to use that have been developed, which makes it hard to pick which τ is useful. We shall, however, see the algorithm reoccur when we analyze adversarial bandits, where we see that the algorithm can prevent itself from 'being tricked' by faulty data.

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 $\rho_a - \mu_a$ 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

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

If we could choose m to minimize this regret, which we find is minimized for

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

Which gives us a regret bounded by

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

This appears to tell us that the algorithm performs poorly if Δ is small, but we have a trivial bound $R_n \leqslant n\Delta$.

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 do not know Δ ahead of time in real applications. The general style for obtaining 'fair' bounds on policies is to first fix a policy,

and then bound the regret over all choices of parameters possible in a situation. In this case, we obtain lower worst case bounds of order $\Omega(n^{2/3})$.

For any n and m, if we let the action gap be $\Delta = (\log n^{4/3m})^{1/2}$, then we find

$$R_n = m\Delta + n\Delta e^{-m\Delta^2/4} = ((4/3m)\log n)^{1/2}[m + n^{2/3}] \in \Omega(n^{2/3})$$

We may even obtain this worst case bound even if we assume the action gap is bounded, say $\Delta \leqslant 2$ (which occurs if our means occur in $[-1, 1]$). Our previous bound works, unless $\Delta^2 \geqslant 4$, in which case $\log n \geqslant 3m$ (where we do not explore nearly enough to expect a good regret bound), and then

$$m\Delta + n\Delta e^{-m\Delta^2/4} \geqslant m\Delta + n\Delta e^{-\log n\Delta^2/12} = \Delta[m + n^{1-(\Delta^2/12)}]$$

We may then set $\Delta = 2$, in which case the regress is bounded below by

$$2m + 2n^{2/3}$$

This lower bound isn't even just asymptotic – there is a universal constant C such that $R_n \geqslant Cn^{2/3}$ over all environments and horizons.

$$\begin{aligned}\inf_{m \leqslant n} \frac{((4/3m)\log n)^{1/2}[m + n^{2/3}]}{n^{2/3}} &= (2/3^{1/2})(\log n)^{1/2} \inf_{m \leqslant n}\left[\frac{m^{1/2}}{n^{2/3}} + \frac{1}{m}\right] \\ &= (2/3^{1/2})(\log n)^{1/2}\frac{1}{n^{4/9}}\left(2^{1/3} + 4^{-1/3}\right)\end{aligned}$$

As $n \to \infty$, this value tends to zero, so that such a C exists. While a very simple algorithm, there are algorithms which increase this rate to $O(n^{1/2})$, which we will soon discuss.

Chapter 2

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 of order $1-\delta$. 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 is useful to increase δ over time, to avoid becoming 'trapped' in suboptimal actions. A standard choice is $\delta_n = 1 + n\log^2 n$, which increases slightly faster than n. If we optimal arm is discredited, an algorithm must pay linear regret. Trying to fix this by adjusting the δ so that this occurs with $1/n$ probability essentially gives us 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 2.1. *Let $X_1, X_2, \ldots$ 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

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

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

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

□

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,

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

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

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

$e^{-\varepsilon^2/2} = 1 - \varepsilon^2/2 + \varepsilon^4/2$ 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 2.2. *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.

The bound above requires us to know the action gaps. But it is fairly easy to obtain a action-gap free bound. Indeed, if $\Delta = \sqrt{|A|C\log n/n}$, then

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

Chapter 3

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. Once we have found the appropriate lower bound for R_n^* over all 1-subgaussian bandits, we will have essentially verified that the upper confidence bound algorithm is optimal for the class of 1-subgaussian environments.

Theorem 3.1. *For any action space A, let $\mathcal{E}$ be the space of 1-subgaussian environments. Then there is a universal constant C such that*

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

To prove this, we apply the High probability Pinsker bound of information theory. Given an environment μ, policy ν, and fixed horizon n, we let $\mathbf{P}_{\mu,\nu}$ be the distribution over histories of length n induced by the environment and policy, and $\mathbf{E}_{\mu,\nu}$ 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.

Lemma 3.2. *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. define $\lambda = \sum_a \mu(a) + \nu(a)$, so that all measures involed are absolutely continuous with respect to λ. Then by the chain rule,

$$D(\mu(a), \nu(a)) = \int \frac{d\mu(a)}{d\lambda} \log\left(\frac{d\mu(a)}{d\nu(a)}\right) d\lambda$$

□

Chapter 4

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

$$A_1, X_1, A_2, X_2, \ldots$$

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 $a_1, a_2, \ldots, a_n$ 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

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

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 4.1. *For any environment $v \in [0,1]^{n \times A}$, 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.

4.1 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.

4.2 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

'new average' = 'old average' + 'step size'['new value' − '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.

4.3 Confidence Intervals for Linear Bandits

s

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.

Before we end this short chapter