Reinforcement Learning

Jacob Denson

October 16, 2016

Table Of Contents

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

Bandit Problems

Imagine you are at a casino, with rows upon rows of slot machines. You of course, being a strategic game player, want to find the slot machine that maximizes your profit from gambling (Casino's do actually adjust the odds of certain slot machines to give certain players the 'feel' of being ahead of the casino, to promote further play.). You hope that you will end up with a profit, but you will also be satisfied, if this is not possible, to find the machine that results in you losing the least amount of money. From the start, you have no idea how any slot machines work, so the only way to determine its reward is to get results from pulling its lever. Be careful though, because pulling a level loses you money! You must balance the act of exploring (learning a machine's distribution) to exploiting (taking advantage of what you know to make the most money).

This is the multi-armed bandit problem. It forms a good problem to begin the attack on reinforcement learning, because it is the simplest type of problem which is mathematically tractable, yet still possesses the exploration-exploitation trade-off, inherent in all problems related to reinforcement learning. It is not just a toy problem, for one can visualize the problem of obtaining relevant data. If we are trying to learn how successful a particular treatment is, we need to ensure that our medical test (which must be performed on infected patients) only risks harming enough people to conclude how successful the test is.

Mathematically, the domain of the multi armed bandits is given by a set A of actions, and, for each action a, a probability distribution of 'rewards' ρ_a over the real numbers, which represents the distribution of the possible values for performing a given action a. 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 simplest case, the action space A is a finite set of discrete actions. Our goal is then to obtain good enough samples of ρ_a to find a best action to take, in the fewest exploratory moves possible.

After a bandit problem has been specified, we must make a decision on actions we want to take? The key 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 from interactions in the environment. A **decision function** for a policy is a sequence of functions $\pi_1, \pi_2, \ldots$, where π_k maps histories of length $k-1$ to a particular action. The policy 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

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

where $(X_k|A_k)$ is independant of $A_1, X_1, \ldots, A_{k-1}, X_{k-1}$. An alternate description of a deterministic policy is a sequence of random variables $A_1, A_2, \ldots$ such that A_k is $\sigma(A_1, X_1, \ldots, A_{k-1}, X_{k-1})$ measurable, and

$$(X_k|A_1, X_1, \ldots, X_{k-1}, A_k) \sim \rho_{A_k}$$

This measure theoretic definition leads to a precise definition of a stochastic random policy, which chooses actions based on the information in some filtration $\{\mathcal{F}_k\}$. A **stochastic policy** is $A_1, A_2, \ldots$, 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 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, let

$$\mu_a = \mathbf{E}(X_k|A_k = a) = \int_{\mathbf{R}} d\rho_a$$

In the finite case, we may define $\mu^* = \max \mu_a$, which is the reward taken for choosing the 'best' 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 the distributions of each action, 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 we do not want to maximize 'good' decisions, but to minimize the number of 'bad' decisions we make. Define the regret function

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

In a good policy, the regret should have a low order of growth over time. It turns out that it is perfectly feasible to have sublinear ($o(n)$) growth, and we shall soon describe general algorithms whose order of regret is $O(\log n)$.

First, we find a useful formula. Define a new random variable

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

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

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

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

We begin by attempting to estimate the expected reward of an action. The law of large numbers tells us that, provided we sample an action infinitely many times, the random variables

$$\widehat{\mu}_a(n) = \frac{1}{T_a(n)}\sum_{k=1}^{n}\mathbf{I}\{A_k = a\}X_k$$

converges to μ_a as $n \to \infty$ almost surely. There is a computational trick for keeping track of these estimates without having to store the indicator function. We have

$$\widehat{\mu}_a(n+1) = \frac{T_a(n)\widehat{\mu}_a(n) + \mathbf{I}\{A_{n+1} = a\}X_{n+1}}{T_a(n+1)}$$

and this gives a recurrence relation meaning we need only keep track of $T_a(n)$ and $\widehat{\mu}_a(n)$, for the rest of the data is given to us sequentially. This may seem to solve our problem completely, for our change in regret will then converge to zero once we begin selecting optimum actions. But, if we start selecting optimal actions, then we won't necessarily sample all actions infinitely often, which implies that our estimates will not converge. Thus we must form a balance of acting in the greedy way, choosing an action with the highest sample average, and acting in an exploratory way.

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. A smarter policy, the ε greedy method, attempts to fix this issue by guaranteeing convergence at the cost of complete optimality.

1.2 The Epsilon-Greedy Method

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

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

No action has non-zero probability, so $\widehat{\mu}_a(k)$ converges to μ_a for each a almost surely. This means that, in the limit, the ε-greedy strategy will act optimally $\varepsilon + (1-\varepsilon)/|A|$ percent of the time. If we decrease the value of ε, our policy will act more optimally in the limit, but it will take longer for $\widehat{\mu}_a(k)$ to converge to $\widehat{\mu}_a(k)$. The expected change in regret for π converges in the limit to

$$\frac{\varepsilon}{|A|} \sum_{a \in A} \Delta_a$$

as the length of the experience extends to infinity, and hence we see that $R_n \sim n\varepsilon \left(\sum \Delta_a\right)/|A|$, and our regret increases linearly, which has a low rate provided most actions have similar average rewards.

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 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, our regret converges to

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

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. This means unless we can find an optimal value of τ for softmax by experimentation, it is hard to recommend softmax over an ε greedy algorithm do to the more complicated definition of softmax.

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 choosen the action a' corresponding to the maximum estimated mean $\widehat{\mu}_a$. The rate of growth of the expected regret then becomes the difference

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

Now 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 obtain a bound for the expected regret, of the form

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

The expected regret is vary 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

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

which is minimized for

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

Which gives us a regret bounded by

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

It may appear that we have attained logarithmic regret, but this regret depends on both Δ and n. If we choose m based only on n (and then choose Δ which gives the worst expected regret), then we can attain $O(n^{2/3})$ regret, but no better (a worst-case bound, rather than a problem/distribution dependant bound). There are algorithms which increase this rate to $O(n^{1/2})$, which we will soon discuss. For very small values of Δ, the bound above

becomes very large, but remember that any policy on a two armed bandit has regret bounded by $n\Delta$.

Note that for $\Delta \leqslant \sqrt{n}$, $n\Delta \leqslant n^{3/2}$, and the second bound obtains a local maximum at $\Delta = \sqrt{4e/n}$, where our bound is $4\sqrt{n/e}$. We also have a possible upper bound at $\Delta = \sqrt{n}$, where our bound is

$$\frac{4(1 + 2\log(n) - \log(4))}{\sqrt{n}} = o(\sqrt{n})$$

Hence R_n is $O(\sqrt{n})$ if action gaps are small enough with respect to n.

1.4 The Upper Confidence Bound 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. If $X_1, \ldots, X_n$ 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 decrease δ over time as our estimates become more accurate. A standard choice is

$\delta_n = \frac{1}{1+n\log^2 n}$, which decreases slightly faster than $\frac{1}{n}$. 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 $X_1, X_2, \ldots$ 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 $\mathbf{E}(T_a(n))$,

for a non-optimal action a. Note that for a fixed ε,

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

To bound these indicator functions, we first define $\widehat{\mu_{a,s}}$ to be the s'th estimate of μ_a (i.e. the value of the estimates $\widehat{\mu_a}(n)$ for $T_a(n) = s$).

$$\begin{aligned}
&\mathbf{E}\left[\sum_{k=1}^{n} \mathbf{I}\left(\widehat{\mu^*}(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \leqslant \mu^* - \varepsilon\right)\right] \\
&= \sum_{k=1}^{n} \mathbf{P}\left(\widehat{\mu^*}(k-1) + \sqrt{\frac{2\log 1/\delta_k}{T_a(k-1)}} \leqslant \mu^* - \varepsilon\right) \\
&\leqslant \sum_{k=1}^{n} \sum_{s=1}^{k} \mathbf{P}\left(\widehat{\mu_s^*} + \sqrt{\frac{2\log 1/\delta_k}{s}} \leqslant \mu^* - \varepsilon\right) \\
&\leqslant \sum_{k=1}^{n} \delta_k \sum_{s=1}^{n} e^{-s\varepsilon^2/2} \\
&\leqslant \sum_{k=1}^{n} \delta_k \int_0^n e^{-s\varepsilon^2/2} ds \\
&\leqslant 5/\varepsilon^2
\end{aligned}$$

where we have used the fact that $1/\delta_k = 1 + k\log^2 k$. 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.

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

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

1.5 Lower Bounding Regret

It turns out that, if we measure a bandit policy by its worst case situation, the UCB algorithm has the best asymptotic bound possible over all situations. That is, we are measuring a policy according to the **minimax regret**.

1.6 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

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

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.

Chapter 2

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