CS234 Notes - Lecture 11,12
Exploration and Exploitation

Anchit Gupta, James Harrison, Emma Brunskill

June 14, 2018

1 Introduction

We have talked previously about desiderata for reinforcement learning algorithms. In particular, we would like to achieve good empirical performance in addition to asymptotic convergence. In many real applications such as education, health care, or robotics, asymptotic convergence rates are not a useful metric for comparing reinforcement learning algorithms. To achieve good real world performance, we want rapid convergence to good policies, which relies on good, strategic exploration.

Online decision-making involves a fundamental tradeoff between exploitation and exploration. Exploitation is making the best possible decision (by maximizing future reward), and exploration involves taking an immediately suboptimal action to gather information. While the suboptimal action will necessarily involve a reduced amount of reward in the immediate future, it may allow you to learn better strategies that enable policy improvement in the long term.

2 Multi-Armed Bandits

We will begin by discussing exploration in the context of multi-armed bandits (MABs), as opposed to full MDPs. An MAB is a tuple $(\mathcal{A}, \mathcal{R})$, where $\mathcal{A}$ denotes a set of actions, and $\mathcal{R}$ is a collection of probability distributions, where each action has a corresponding distribution over rewards $\mathcal{R}^a(r) = \mathbb{P}(r|a)$. At each time step, an agent selects an action a_t. As in MDPs, the agent aims to maximize cumulative reward. This setting does not have the state transitions of an MDP, and thus there is no notion of delayed rewards or consequences.

Let $Q(a) = \mathbb{E}[r|a]$ denote the true expected reward for taking action a. We will consider algorithms that estimate $\hat{Q}_t(a) \approx Q(a)$. The value is estimated by Monte-Carlo evaluation,

$$\hat{Q}_t(a) = \frac{1}{N_t(a)} \sum_{t=1}^{T} r_t \mathbf{1}(a_t = a) = \hat{Q}_{t-1}(a) + \frac{1}{N_t(a)} \left(r_t - \hat{Q}_{t-1}(a) \right) \tag{1}$$

where $N_t(a)$ is the number of times action a has been taken at time t. The second equality is useful for computing estimates $\hat{Q}_t$ incrementally.

The **greedy** algorithm selects the action with the highest estimated value, $a_t^* = \arg\max_{a \in \mathcal{A}} \hat{Q}_t(a)$. However, the greedy approach may lock onto a suboptimal action forever, as an exercise try constructing such a bandit problem. Like in MDPs, we can also take a (fixed) **ϵ-greedy** algorithm selects a greedy action with probability $1 - \epsilon$, and a random action with probability ϵ. Another algorithm is **decaying ϵ_t-greedy**, in which ϵ_t decays according to some schedule.

A simple alternative to ϵ-greedy based approaches is **optimistic initialization** which initializes $\hat{Q}_0(a)$ for all $a \in \mathcal{A}$ to be some large value greater than the true value $Q(a)$. In other words, we start off "wildly optimistic" about all action choices. On each step, we can use a greedy (or ϵ-greedy) approach for selecting the action with the highest $\hat{Q}_t(a)$. Since the true rewards are all less than our initial estimates, actions that have been explored will see decreases in their estimated $\hat{Q}$ values, thus encouraging exploration among unexplored actions that still have large $\hat{Q}$ values. Thus, all actions will be tried at least once, and likely many more times. Furthermore, we can initialize $N_0(a) > 0$ to trade off how fast the optimistic initializations converge towards the true values.

2.1 Regret

These exploration strategies naturally give rise to the question of which metric we are using to compare them. Possible metrics include empirical performance (although this is environment dependent), asymptotic convergence guarantees, finite sample guarantees, or PAC guarantees. The standard in the MAB literature is typically **regret**. We will define regret and related quantities.

- Action-value $Q(a) = \mathbb{E}[r|a]$
- Optimal Value $V^* = Q(a^*) = \max_{a \in \mathcal{A}} Q(a)$
- Gap $\Delta_a = V^* - Q(a)$
- Regret $l_t = \mathbb{E}[V^* - Q(a_t)]$
- Total regret $L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} (V^* - Q(a_\tau))\right] = t \cdot V^* - \mathbb{E}\left[\sum_{\tau=1}^{t} Q(a_\tau)\right]$

Minimizing total regret is thus equivalent to maximizing cumulative reward. If we define the count $\bar{N}_t(a)$ as the expected number of selections of action a, we can see that the total regret is a function of the gap and the counts,

$$L_t = \mathbb{E}\left[\sum_{\tau=1}^{t} (V^* - Q(a_\tau))\right] \tag{2}$$

$$= \sum_{a \in \mathcal{A}} \mathbb{E}\left[N_t(a)\right] (V^* - Q(a)) \tag{3}$$

$$= \sum_{a \in \mathcal{A}} \bar{N}_t(a) \Delta_a. \tag{4}$$

A good algorithm will ensure that the counts are small for large gaps. However, the gaps are not known in advance, and must be learned by interacting with the MAB.

2.2 Regret Bounds

It is desirable to guarantee that the regret of some algorithm can be quantified and bounded. There are two types of regret bounds: problem dependent and problem independent. Problem dependent regret bounds are a function of the number of times each arm is pulled and the gaps. Problem independent bounds are strictly functions of T, the total number of steps the algorithm operates for.

An algorithm that explores forever or chooses a suboptimal action forever will experience linear regret. It is desirable therefore to achieve sublinear regret. The regret bounds of the previously discussed algorithms are as follows:

- **Greedy:** linear total regret

- **Constant ϵ-greedy:** linear total regret
- **Decaying ϵ-greedy:** Sublinear regret but schedule for decaying ϵ requires knowledge of gaps
- **Optimistic initialization:** Sublinear regret if initial values are sufficiently optimistic, else linear regret.

To give a sense of how hard the problem is, it is useful to establish a lower bound on regret. Generally, the performance of any algorithm is determined by the similarity between the optimal arm and other arms. Hard problems will have similar arms with slightly different means. This can be described by the gaps Δ_a and the similarity in distributions (via the KL divergence), $KL(\mathcal{R}^a \| \mathcal{R}^{a^*})$. Then, we may establish a bound on the asymptotic total regret.

Theorem 1 (Lai and Robbins, 1985). *Any algorithm on a MAB has an asymptotic lower bound on total regret of at least*

$$\lim_{t \to \infty} L_t \geq \log t \sum_{a | \Delta_a > 0} \frac{\Delta_a}{KL(\mathcal{R}^a \| \mathcal{R}^{a^*})}.$$

2.3 Optimism in the Face of Uncertainty

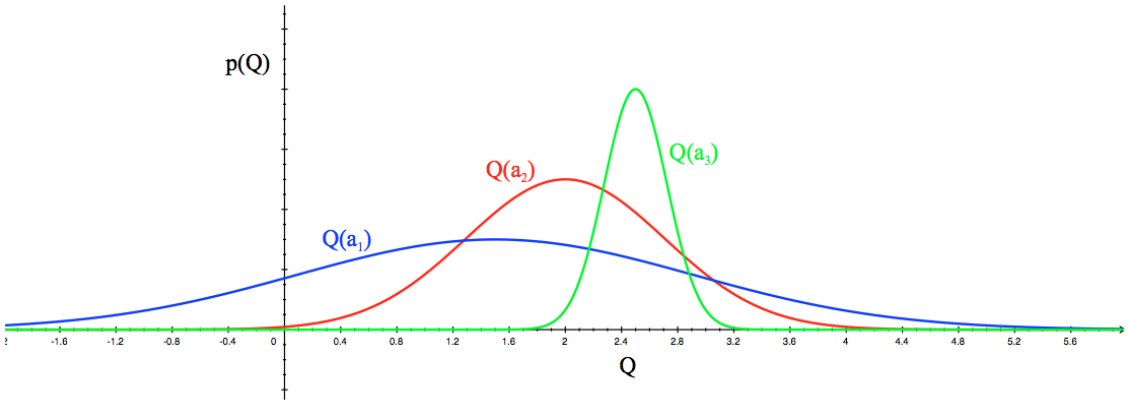

Figure 1: Reward distributions for actions a_1, a_2, a_3

Consider Figure 1, showing estimated distributions for a collection of actions. Which action should we choose? The principle of optimism in the face of uncertainty says that we should bias our choice toward actions which *may* be good. Intuitively, this will lead to either learning that the action does in fact result in high reward, or we learn that the action is not as good as we may have hoped, and we have learned valuable information about our problem.

This approach gives rise to the **Upper Confidence Bound** algorithm, which proceeds as follows. First, we estimate an upper confidence $\hat{U}_t(a)$ for each action value, such that $Q(a) \leq \hat{Q}_t(a) + \hat{U}_t(a)$ with high probability. This depends on the number of times action a has been selected. Then, we select actions to maximize the upper confidence bound

$$a_t = \arg\max_{a \in \mathcal{A}} \left\{ \hat{Q}_t(a) + \hat{U}_t(a) \right\} \tag{5}$$

This can be derived via Hoeffding's inequality.

Theorem 2 (Hoeffding's Inequality). *Let $X_1, \ldots, X_t$ be i.i.d. random variables in $[0,1]$, $\bar{X} = \frac{1}{t}\sum_{\tau=1}^{t} X_\tau$ be the sample mean, and u denote a constant. Then,*

$$\mathbb{P}[\mathbb{E}[X] > \bar{X}_t + u] \leq \exp(-2tu^2).$$

Applying the above to the bandit problem, we get

$$\mathbb{P}[Q(a) > \hat{Q}_t(a) + U_t(a)] \leq \exp(-2N_t(a)U_t(a)^2). \tag{6}$$

Choosing a probability p that exceeds the true upper confidence bound, we get

$$\exp(-2N_t(a)U_t(a)^2) = p \tag{7}$$

$$U_t(a) = \sqrt{\frac{-\log p}{2N_t(a)}}. \tag{8}$$

We will reduce p as we observe more rewards. In particular, choosing $p = t^{-4}$ yields the **UCB1** algorithm,

$$a_t = \arg\max_{a \in \mathcal{A}} \left\{ Q(a) + \sqrt{\frac{2\log t}{N_t(a)}} \right\}, \tag{9}$$

which ensures asymptotically optimal action selection i.e it matches the [1] lower bound upto constant factors.

Theorem 3 (Auer et al., 2002, [2]). *The UCB algorithm achieves asymptotic total regret*

$$\lim_{t\to\infty} L_t \leq 8\log t \sum_{a|\Delta_a>0} \Delta_a.$$

2.4 Bayesian Bandits

With the exception of assumption boundedness of rewards, we have so far made no assumptions about the reward distribution $\mathcal{R}$. **Bayesian bandits** exploit prior knowledge of rewards to compute the probability distribution of rewards in the next time step i.e $p[\mathcal{R}]$ where $h_t = (a_1, r_1, \ldots, a_{t-1}, r_{t-1})$, and then use this posterior to guide action selection. For example, consider a Bayesian approach to UCB. We will assume the reward distribution is Gaussian, $\mathcal{R}^a(r) = \mathcal{N}(r; \mu_a, \sigma_a^2)$, and our posterior over μ_a and σ_a^2 may be computed by

$$p[\mu_a, \sigma_a^2 | h_t] \propto p[\mu_a, \sigma_a^2] \prod_{t|a_t=a} \mathcal{N}(r_t; \mu_a, \sigma_a^2). \tag{10}$$

We can pick the action that maximizes the standard deviation of $Q(a)$,

$$a_t = \arg\max_{a \in \mathcal{A}} \left\{ \mu_a + c\frac{\sigma_a}{\sqrt{N(a)}} \right\} \tag{11}$$

An alternative approach is **probability matching**, which selects an action a according to the the probability that a is the optimal action.

$$\pi(a|h_t) = \mathbb{P}(Q(a) > Q(a'), \forall a' \neq a \mid h_t). \tag{12}$$

This is optimistic in the face of uncertainty, as uncertain actions have a higher probability of being the best action. However, this probability may be difficult to compute analytically.

An approach to probability matching is **Thompson sampling**, in which a reward distribution $\hat{\mathcal{R}}$ is sampled from the posterior for each $a \in \mathcal{A}$. Then, the action-value function $\hat{Q}(a) = \mathbb{E}[\hat{\mathcal{R}}^a]$ may be

computed, and we select the action with the highest $\hat{Q}(a)$. This approach achieves the Lai and Robbins lower bound [1], and can be extremely effective in practice.

As an example in the case of Bernoulli bandits it can be shown that if the prior is a Beta distribution then the required posterior distribution is also a Beta distribution. Hence we can start off from a beta prior over the arms play according to this in the next time step and subsequently update the parameters of the posterior. More concretely if S_i is the number of 1s observed for arm i so far and F_i is the number of 0s observed the following pseduo-code 1 implements the algorithm.

Algorithm 1 Thompson Sampling for Bernoulli MAB using Beta priors

1: **for** $i = 1, 2, \ldots$ **do**
2: For each arm $k = 1, ..N$, independently sample $\theta_k \sim Beta(S_k + 1, F_k + 1)$
3: Play arm $a_i = arg\ \max_k \theta_k$
4: Observe r_i and update S_{a_i}, F_{a_i} accordingly

A similar approach can be applied in the case of gaussian MAB with unit variance. It can be seen that that after time t the posterior for arm k equals $N(\mu_k, \frac{1}{S_k+F_k+1})$ where μ_k is the empirical average reward. As an exercise try to verify the above claim.

2.5 PAC bandits

All the algorithms described above seek to achieve regret bounds in terms of T. But this does not give us an idea of the types of mistakes the algorithm is making, it could be making large mistakes infrequently or smaller ones frequently. In many applications we may care about the bounding the number of large mistakes.

Formally a PAC algorithm guarantees that the algorithm will choose an action whose values is ϵ optimal i.e $Q(a) \geq Q(a^*) - \epsilon$ with probability atleast $1 - \delta$ on all but a polynomial number(usually in terms of ϵ, δ, N) of time steps. There exist variants of the optimism under uncertainty and Thompson sampling algorithms with such PAC guarantees.

3 Information State Search

The fundamental conflict between exploration and exploitation stems from the fact that exploration gains information which might help in the future but is sub optimal as of now. If we are able to quantify this "Value of Information" in terms of how much reward you should be prepared to pay for that information we can much more efficiently balance the exploration-exploitation trade-off. As a concrete example refer the seismologist example in the slides.

3.1 Information State Space

So far we viewed MAB's as a simple fully observable MDP with a single state.
Main Idea: Frame a MAB problem as a partially observable MDP where the hidden state is the actual mean reward of each arm. Actions as before correspond to pulling the arms. The observations we get are the rewards sampled from the hidden state. Hence finding a optimal policy for this POMDP gives us and optimal bandit algorithm. In other words and MAB instance can be reduce to an instance of POMDP planning.

A main idea from POMDP planning is that of a belief state $\tilde{s}$ which can be though of as an information state in our context. This is a posterior over the hidden state of the POMDP i.e the true mean rewards

in our case. $\tilde{s}$ is a statistic computed using the history i.e $\tilde{s_t} = f(h_t)$. Each action and it consequential observation (reward) results in a probabilistic transition to a new state $\tilde{s_{t+1}}$ in the information(belief) state space. This turns out be an MDP over the augmented information state space.

3.2 Bernoulli Bandits

In the case of simple Bernoulli bandits the information state is just the counts of the 1,0 rewards for each arm. As these counts are unbounded we now have an infinite MDP over these information states. This MDP can be solved by RL for eg. using model free techniques like Q-Learning or Bayesian model based RL. This approach is known as Bayes-adaptive RL where we seek to find the Bayes optimal exploration/exploitation trade-off given the prior i.e select the action that maximizes expected reward given current information.

The Thompson sampling algorithm described earlier for Bernoulli Bandits can be thought of in these terms with the the S_k, F_k values for each arm representing the information state and the updates to the them the transitions.

3.2.1 Gittins Index

The above Bayes-adaptive MDP can be solved using Dynamic programming and this solution is called the Gittins index. The exact solution to this is typically intractable due to the size of state space but recently simulation based search has been applied to this problem to get very good results. You will learn about such simulation methods in lecture 14.

4 Application to MDP's

All of the above methods can be extended too apply to full fledged MDP's

4.1 Optimistic Initialization

In the model free RL case systematic exploration can be very easily encouraged using optimistic initialization of the value function. We can initialize $Q(s, a)$ to $\frac{R_{\max}}{1-\gamma}$ and then run algorithms like Q learning, Sarsa, Monte-carlo etc. The high initial values ensure that each s, a pair is explored a number of times to get a better estimate of its true value.
A similar idea works in he case of model based methods where we construct and optimistic model of the MDP and initialize transitions to go to the terminal state with the maximum reward. The RMax algorithm [3] is an example of such an algorithm.

4.2 UCB: Model Based

Similar to the MAB case we can chose a action which maximises the upper confidence bound on the available actions.

$$a_t = arg \max_A Q(s_t, a) + U_1(s_t, a) + U_2(s_t, a)$$

where U_1 is the uncertainty in the policy evaluation which is easy to quantify and the second term arises from the uncertainty in the policy improvement step and is typically hard to compute.

4.3 Thompson Sampling: Model Based

As before this implements probability matching i.e

$$\pi(a|h_t) = \mathbb{P}(Q(s,a) > Q(s,a'), \forall a' \neq a \mid h_t). \tag{13}$$

We can use Bayes law to compute the posterior $\mathbb{P}[P, R|h_t]$, and then sample a MDP from this distribution, solve it using a planning algorithm and act accordingly.

4.4 Information State Search

As in the MAB case we can construct a augmented MDP with the information state appended to the MDP's state to get an augmented state space to get a Bayes adaptive MDP. Solving this would then give us the optimal exploration/exploitation trade off. However the size of the state space restricts us to using simulation based search methods to solve these.

5 Conclusion

In summary there are a variety of exploration techniques some more principled than others.

- Naive exploration - ϵ-greedy methods
- Optimistic Initialization - A very simple idea that usually works very well
- Optimism under uncertainty - prefer actions with uncertain values eg. UCB
- Probability matching - pick action with largest probability of being optimal eg. Thompson sampling
- Information state space - construct augmented MDP and solve it, hence directly incorporating the value of information.

References

[1] Lai, Tze Leung, and Herbert Robbins. "Asymptotically efficient adaptive allocation rules." Advances in applied mathematics 6.1 (1985): 4-22.

[2] Auer, Peter, Nicolo Cesa-Bianchi, and Paul Fischer. "Finite-time analysis of the multiarmed bandit problem." Machine learning 47.2-3 (2002): 235-256.

[3] Brafman, Ronen I. and Tennenholtz, Moshe. "R-max - a General Polynomial Time Algorithm for Near-optimal Reinforcement Learning" JMLR (2003) 213-231.