MS&E338 Reinforcement Learning Lecture 13 - May 14, 2018

Active Learning Algorithms

Lecturer: Ben Van Roy *Scribe: Jiaxing Geng and Qian Sarah Mu*

1 Review: Active Learning

The variables defined in general online decision problems:

- Action: $x_t \in X$, where X is the action set.
- Observation: $y_t \sim q_\theta(\cdot|x_t)$. y_t is sampled from the distribution q_θ (conditioning on action x_t).
- Reward: $r_t = r(y_t)$.
- Prior: $\theta \sim p$. θ is sampled from the prior distribution p.

We can calculate the expected reward of a model after taking action x:

$$\bar{r}(x, \theta) = \mathbb{E}\left[r(y_t)|x_t = x, \theta\right] = \sum_y q_\theta(y|x_t) r(y)$$

$\bar{r}(x, \theta)$ is widely used to select the best action in the following algorithms.

1.1 Greedy Algorithm

We evaluate the model for each time period. In each period, a natural way to estimate the true model parameter is by taking the expectation value of the prior distribution and we denote this prediction by $\hat{\theta}$. Then, we select an action to maximize the expected reward under this model. After observing y_t, we update the prior distribution p via Bayes rule.

Algorithm 1 Greedy Algorithm(X,p,q,r)

for t=1,2,... **do**
  $\hat{\theta} \leftarrow \mathbb{E}\left[\theta|p\right]$
  $x_t = argmax_{x \in X} \bar{r}(x, \hat{\theta})$
  observe y_t
  $p \leftarrow \mathbb{P}\left\{\theta \in \cdot | p, x_t, y_t\right\}$
end for

1.2 Thompson Sampling

Different from Greedy Algorithm, instead of using the mean of the prior distribution, we sample $\hat{\theta}$ from the prior distribution as the predicted parameter during each time period. Then, we select an action to maximize the expected reward under this model. After observing y_t, we update the prior distribution p via Bayes rule.

Algorithm 2 Tompson Sampling(X,p,q,r)

for t=1,2,... **do**
  sample $\hat{\theta} \sim p$
  $x_t = argmax_{x \in X} \bar{r}(x, \hat{\theta})$
  observe y_t
  $p \leftarrow \mathbb{P}\{\theta \in \cdot | p, x_t, y_t\}$
end for

1.3 Upper Confidence Bound Algorithm(UCB)

We evaluate the model for each time period. In UCB, the action is selected to maximize the function $U_{p,t}(x)$, instead of the expected reward. We define confidence set Θ, which includes all statistically plausible models. $U_{p,t}(x) = max_{\hat{\theta} \in \Theta} \bar{r}(x, \hat{\theta})$. In other words, the result of each period is maximized over both models and actions. After observing y_t, we update the prior distribution p via Bayes rule.

Algorithm 3 UCB(X,p,q,r)

for t=1,2,... **do**
  $x_t = argmax_{x \in X} U_{p,t}(x)$
  observe y_t
  $p \leftarrow \mathbb{P}\{\theta \in \cdot | p, x_t, y_t\}$
end for

2 Review: Beta-Bernoulli Bandit Problem

There are k coins to choose from, each with unknown probability. In each time period, we toss one of the k coins. If the head of the coin shows up, we get 1 dollar as reward. We can formulate this problem as the following:

- Action: $x_t \in X = \{1, 2, ..., K\}$. The action set is described by which coin we choose.
- Observation: $y_t \sim q_\theta(\cdot | x_t)$. $q_\theta(y|x) = \theta_x$ if $y = 1$, and $q_\theta(y|x) = 1 - \theta_x$ if $y = 0$.
- Reward: $r_t = r(y_t)$. $r(y) = 1$ if $y = 1$ and $r(y) = 0$ if $y = 0$.
- Prior: $p(\theta) \propto \Pi_{k=1}^{K} \theta_k^{\alpha_k - 1} (1 - \theta_k)^{\beta_k - 1}$. We assume that each coin is not correlated with the other coins and its prior distribution is beta distribution.

2.1 Bayes Rule in Beta-Bernoulli Bandit

In all kinds of algorithms showed above, we change the prior distribution based on what we observe during each time period. For beta distribution, the update formula of applying Bayes rule is simple. $\alpha_{x_t} \leftarrow \alpha_{x_t} + 1$ if $y_t = 1$ and $\beta_{x_t} \leftarrow \beta_{x_t} + 1$ if $y_t = 0$.

2.2 UCB Algorithm in Beta-Bernoulli Bandit

People often use two kinds of UCB algorithms: "Bayes UCB" and "UCB1."

2.2.1 Bayes-UCB

The UCB function $U_{p,t}(x)$ is defined as

$$U_{p,t}(x) = quantile(\theta_x, 1 - \frac{1}{t})$$

where t is the time period.

2.2.2 UCB1

UCB1 is ***asymptotically*** optimal, which is usually served as a sanity check. The UCB function $U_{p,t}(x)$ is defined as

$$U_{p,t}(x) = \mathbb{E}_\theta \left[\bar{r}(x, \theta)\right] + \gamma \sqrt{\frac{2ln(t)}{t_x}}$$

where t is the time period and t_x is the number of times we have selected coin x.

The intuition of $ln(t)$ in $U_{p,t}(x)$ is the following: we want to slightly increase the confidence interval over time, such that even though some coins has low expected rewards at the start, we are still able to try them out some time in the future. The intuition of t_x is the denominator is that we want to shrink the confidence interval after trying coin x. Besides, UCB1 does not perform as well as Bayes-UCB, if we don't tune the hyper-parameter γ.

3 Shortest Path Problem

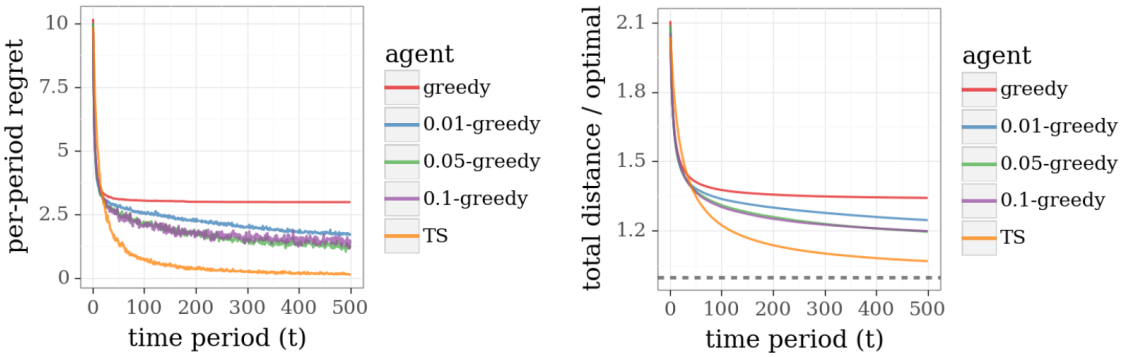
Given a graph (V,E) with vertices V and edges E, we are trying to find the shortest path from origin to destination. The problem can be formulated as the following:

- Action: $x_t \in X$. $x_t = (e_1, e_2, ..., e_n)$, which is a sequence of edges we traverse. X is all possible paths from origin to destination.
- Observation: $y_t \sim q_\theta(\cdot|x_t)$. $y_t = (d_1, d_2, ..., d_n)$, which includes the delay of each edge following action x_t. Conditioned on the path, q_θ imposes a log-normal distribution with mean θ_e for each edge e in the path, and the distribution is independent across edges.
- Reward: $r_t = r(y_t) = -\sum_{i=1}^{n} d_i$.
- Prior: $\theta_e \sim p_e$, for all $e \in E$. The prior distribution p_e is a log-normal distribution, with parameter μ_e and σ_e^2.

3.1 Shortest Path Problem Overview

The prior distribution update is relatively simple using log-normal distribution, and the formula can be found in Tompson Sampling Tutorial(Equation 4.3). It is also useful notice that observation y_t offers much richer information that the reward, as it provides all the delays over edges. One possible issue is raised when considering the size of action set. The number of decisions we can grow exponentially with the number of edges and the action set is huge in this setting. It seems that it would take a long time to converge. However, in the shortest path problem, since a lot of the paths will traverse the same edges, the actions are related to each other, which greatly shortens the computation time.

The following graph shows a comparison of different algorithms, including greedy, ϵ-greedy and TS. From the graph, TS performs the best and ϵ-greedy is slightly better than greedy.



(a) regret **(b)** cumulative travel time vs. optimal

Figure 1: Comparison of Different Algorithms in Shortest Path Problem

3.2 UCB Algorithm for Shortest Path Problem

For UCB, we have to decide which confidence set we want to use. In shortest path problem, where $ln(\theta) \sim \mathcal{N}(\mu, \Sigma)$, we can identify the shape of confidence set by looking at the level set of normal distribution, which is an ellipsoid. However, jointly optimizing the models on this confidence set and the actions leads to an NP-hard problem. Thus, we need a different approach to choose a confidence set – a set that is very likely to contain the true model, and rule out the ones that are not likely.

The alternative approach is that, for each edge, based on the distribution, we impose upper and lower confidence bars, such that anything in the intersection is statistically plausible. It would then make the joint optimization problem very easy. We can choose the lower bound for each edge (being optimistic), then choose the path that is the shortest. This approach works very fast, but it is not statistical efficient, since it requires many more trials that grows with number of edges. In general, using the "correct" confidence sets can often lead to computational intractability for UCB methods as they need to jointly optimize over actions and model parameters. To deal with this, people sometimes use box-shaped confidence sets, which can cause a lot of problems as we will explain.

UCB can be easily beaten by TS when the wrong shape of the confidence set is used or if the confidence set is too loose. Aditionally, TS is usually more computationally efficient. But if the right confidence set is used, UCB typically performs slightly better than TS.

3.3 Example Comparing Total UCB and Separate UCBs

In this example, we want to maximize the accumulation of rewards and there are two separated paths that we can select. On the top path, we have learned everything we want to learn: the reward is 1 from the origin to the next state and all the following rewards are 0. For the bottom path, we are uncertain about the rewards for the first τ edges, which we assume are independent with mean 0 and standard deviation $\frac{\epsilon}{\sqrt{\tau}}$. The rewards are 0 after the first τ steps.

We would like to figure out the rewards of two paths(actions). There are two approaches to UCB: Total UCB and Separate UCBs. Conceptually, Total UCB corresponds to the ellipsoid confidence set and Separate UCBs corresponds to the "rectangular" confidence set.

3.3.1 Total UCB:

Reward along the top path = 1.
Reward along bottom = $0 \pm \sqrt{\tau(\frac{\epsilon}{\sqrt{\tau}})^2} = \pm\epsilon$. Here, we look at the joint distribution of τ independent edges so the variance of their sum is the sum of their variances.

3.3.2 Separate UCBs:

Reward along the top path = 1.
Reward along the bottom path = $\pm\tau\frac{\epsilon}{\sqrt{\tau}} = \pm\epsilon\sqrt{\tau}$.
Compare the rewards of two path in this case. We can propose to explore the second path if $1 < c\epsilon\sqrt{\tau}$. When τ is really large, the algorithm tends to always select the bottom path, which is not desirable. This issue is named as temporal incoherence, which occurs when we try to bound each edge individually. Similar problem occurs when there is a large fan-out with stochastic transition – there are many possible next nodes from one node. This effect is called spatial incoherence.

4 Episodic MDP

MDP is described as M = (S, A, R, P, ρ)
UCRL: UCB algorithm for RL. It suffers from all the UCB problems.
PSRL: Thompson Sampling Algorithm for RL.

Let p denote the prior on the MDP.

Algorithm 4 PSRL Algorithm(p)

for l=1,2,... **do**
  sample $\widehat{M} \sim p$
  compute optimal $\hat{\pi}$ for $\widehat{M}$
  observe s_0
  for t=0,1,2,...,$\tau - 1$ **do**
    $a_t = \hat{\pi}(s_t)$
    observe r_{t+1}, s_{t+1}
  end for
  update p using Bayes rule, conditioning on ($\hat{\pi}$,s_0,a_0,r_1,s_1,...,s_τ)
end for

The algorithm above looks very similar to Thompson Sampling. Indeed, it is a special case of Thompson Sampling. In the table below, we can find the one to one correspondence between our general setting and MDP.

General Setting	MDP
x_t	π_l
θ	M
y_t	(s_0,a_0,r_1,s_1,...,s_τ)
$r(y_t)$	$\sum_{t=1}^{\tau} r_t$