Statistical Techniques in Robotics (16-831, S22) Lecture #14 (Monday, March 14)

Sequence Feedback Learning Problems

Lecturer: Kris Kitani *Scribes: Seth Karten, Siva Kailas (Group 4)*

1 Review

In the last lectures, we covered Thompson Sampling and EXP3/EXP4. The Thompson Sampling method maintains a running estimate of the prior distribution hyperparameter by observing the rewards, and selects the best arm by sampling from the estimated posterior distribution. EXP3 addresses context-free bandits in an adversarial environment, and EXP4 addresses contextual bandits in an adversarial environment. In this lecture, we will cover a review of the learning problems we have covered so far in class. We will compare and contrast the previous methods, as well as understanding the key differences between supervised and reinforcement learning. We will introduce sequential decision-making, including Markov Decision Processes and the mathematics behind it.

1.1 Thompson Sampling

Thompson sampling can be broken down into a 2-step high-level strategy as shown below.

1. Maintain a running estimate of the prior distribution hyperparameter by observing the rewards
2. Select the best arm by sampling from the estimated posterior distribution

We use the version of Thompson sampling that provides simple update rules. The algorithm for the specific variant of Thompson sampling which uses a Bernoulli likelihood function, Beta prior distribution (and posterior distribution), and the derived posterior update is shown below.

Algorithm 1 Bern-Beta Thompsons Sampling

for $t = 1, \ldots, T$ **do**
  $\theta_k \sim p(\theta; \alpha_k, \beta_k) \, \forall k$
  $a_{\hat{k}}^{(t)} = \arg\max_k \mathbb{E}_{p(r|a_k,\theta_k)}[r|a_k, \theta_k]$
  `RECEIVE`$(r^{(t)})$
  $\alpha_{\hat{k}} = \alpha_{\hat{k}} + r^{(t)}$
  $\beta_{\hat{k}} = \beta_{\hat{k}} + 1 - r^{(t)}$
end for

1.2 EXP3 and EXP4

The main idea of Exponential-Weight Update algorithm for Exploration and Exploitation (EXP3) is addressing context-free bandits in an adversarial environment.

Algorithm 2 EXP3($\gamma \in [0, 1]$)

for $t = 1, \ldots, T$ **do**
  $p^t = \frac{w^t}{\sum_k w_k^t}$
  $k \sim \texttt{Multinomial}(p^t)$
  $a^t = a_k$
  $\texttt{RECEIVE}(r^t \in [0, 1])$
  $w_k^{t+1} = w_k^t \exp\{\gamma \frac{r^t}{p_k^t}\}$
end for

The main idea of Exponential-Weight Update algorithm for Exploration and Exploitation with Experts (EXP4) is addressing contextual bandits in an adversarial environment.

Algorithm 3 EXP4($\gamma \in (0, 1], T$)

$w^1 \leftarrow 1 \in \mathbb{R}^N$
for $t = 1, \ldots, T$ **do**
  $\texttt{RECEIVE}(X^t \in \mathbb{R}^{N \times K})$
  $q^t = \frac{w^t}{\|w^t\|} X^t \in \Delta^K$
  $k^t \sim \texttt{Multinomial}(q^t)$
  $\texttt{Receive}(r^t)$
  $\hat{r}^t = \frac{r^t}{q_k^t} \mathbb{I}[k = k^t] \in \mathbb{I}^K$
  $g^t = X^t * \hat{r}^t \in \mathbb{R}^N$
  $w_n^{t+1} = w_n^t \exp\{\gamma * g_n^t\} \ \forall n$
end for

The regret of EXP4 is,

$$R_{\texttt{EXP4}} \leq \sqrt{KT \log N},$$

where K is the number of arms, T is the time, and N is the number of experts.

In `EXP4`, we parameterize the distribution by

$$q^t = \frac{w^t}{\|w^t\|} * X^t,$$

while in `EXP3`, we parameterize the distribution by

$$p^t = \frac{w^t}{\|w^t\|}.$$

`EXP4` uses the matrix of expert advice from N experts to determine the probability over actions and then draw the action from the distribution as the key difference.

This is reflected in the g_n^t terms in `EXP4` versus the reward over inverse probability ratio, $\frac{r^t}{p_k^t}$, in `EXP3`.

2 Review of Learning Problems

Supervised learning is a 'one-shot' feedback method. The key idea is that in this method, one does not have to worry about issues like co-variate shift or temporal credit assignment. Each data-point is independent of all other data-points, i.e., each the state and action does not affect any state and action. Samples are identically and independently distributed. Overall, supervised learning creates a mapping $f : x \to a$.

On the other hand, in reinforcement learning, samples are drawn through interaction. There is a strong correlation between sequential data-points. That is, all samples in a trajectory are correlated where the action affects the next state. The sequence of states and actions creates a single data-point, which is mapped to a reward as a type of 'sequence' feedback. Here, it is important to address covariate shift, temporal credit assignment, and very large "trajectory" spaces.

2.1 Decision-making Problems

Problem	Sampled	Evaluative	Sequential
PWEA	×	×	×
OLC/OMD	✓	Δ	×
MAB	×	✓	×
C-MAB	✓	✓	×
RL	✓	✓	✓
IL	✓	✓	Δ

PWEA (e.g., weight majority algorithm) is neither sampled, evaluative, nor sequential. The expert advice was from expert advice (0 or 1), not sampled. The actions were discrete and finite. It was not evaluative, since we either won or lost the race (for example). It was also not sequential, since we did not get a payout across multiple races.

OLC / OMD is sampled, sometimes instructive and sometimes evaluative, and not sequential. The sampled feedback is, for example, a vector of observations. There is also a vector of paramters in continuous space. Our online linear classification was set up as instructive (observation vector is either class 0 or 1). Using the Hinge loss, this became evaluative. Again, with restarting after each state, it is not sequential.

In both PWEA and OLC/OMD, the loss function was fully observed. In the following multi-arm bandit problems, the loss is no longer fully observed. The MAB problem was not sampled (exhaustive), was evaluative, and was not sequential. The feedback was the reward from a stochastic arm function, so the feedback was completely evaluative.

The (Contextual) C-MAB problem was sampled, was evaluative, and was not sequential. The information was not exhaustive, it was sampled from continuous space. The feedback was evaluative. However, the (noisy) reward function was only partially observed, so we could only update one parameter or arm at a time.

Reinforcement learning is sampled, evaluative, and sequential. This implies that one must reason about the impact of decisions on the entire sequence, including future states within a sequence.

Imitation learning is sampled, evaluative, and sequential. However, we can convert the sequence

into a one-shot problem.

3 Sequential Decision-Making

In sequential problems, one obtains a sequence of rewards and updates the future predictor, which we will call a value function, and then update an action predictor, which we will call a policy.

3.1 Markov Decision Processes

A Markov Decision Process (MDP) is defined by the 7-tuple $(\mathcal{S}, \mathcal{A}_{\mathcal{S}}, p(s'|s,a), r(s',s,a), p_0(s), \pi(a|s), \gamma)$ where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $p(s'|s,a)$ is the state transition dynamic, $r(s',s,a)$ is the reward function, $p_0(s)$ is the state prior, $\pi(a|s)$ is the policy, and γ is the discount factor.

In a traditional MDP factorization, we have,

$$p(s_0, s_1, a_1, \ldots, s_T, a_T) = p_0(s_0) \prod_t p(s_{t+1}|s_t, a_t) p(a_t|s_t).$$

The reward may be factorized in the following ways (from most complex relation to simplest),

$$r(s_0, a_0, s_1, a_1, \ldots, s_T, a_T) = r(s_0, a_0, s_1) + r(s_1, a_1, s_2) + \ldots$$

$$r(s_0, a_0, s_1, a_1, \ldots, s_T, a_T) = r(s_0, a_0) + r(s_1, a_1) + \ldots$$

$$r(s_0, a_0, s_1, a_1, \ldots, s_T, a_T) = r(s_0) + r(s_1) + \ldots$$

3.2 Grid World Example

3.3 Mathematics of the MDP

The state value function is the total expected return of a trajectory starting in state s under policy π,

$$V^{\pi}(s) = \mathbb{E}_p[r_0 + r_1 + r_2 + \ldots | s_0 = s],$$

which yields the summation of all future rewards or the expected return. The probability distribution from the expectation comes from the MDP given the first-order Markov assumption,

$$p(s_0, a_0, s_1, a_1, \ldots) = p_0(s_0) p(s_1|s_0, a_0) p(a_0|s_0) p(s_2|s_1, a_1) p(a_1|s_1) \ldots$$

The return from the value function depends on the horizon. With an infinite horizon, there is an infinite return, which is hard to deal with,

$$V^{\pi} = \mathbb{E}_p[r_0 + r_1 + r_2 + \ldots | s_0 = s].$$

For a finite horizon, the return is,

$$V^{\pi} = \mathbb{E}_p[r_0 + r_1 + r_2 + \cdots + r_T | s_0 = s].$$

An infinite horizon discounted return is,

$$V^{\pi} = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0 = s].$$

The state-action value function, on the other hand, is the total expected return of a trajectory starting in state s and taking action a under policy π.

$$Q^{\pi}(s, a) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a_0 = a]$$

There is a relationship between the state value function $V^{\pi} = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0 = s]$ and state-action function $Q^{\pi}(s, a) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a_0 = a]$. We can derive it by starting with the definition of V^{π} as shown below.

$$V^{\pi}(s) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0 = s] = \mathbb{E}_p \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right]$$

From this, we apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ as shown below.

$$V^{\pi}(s) = \mathbb{E}_p \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right] = \sum_{s_{1:\infty}, a_{0:\infty}} p(s_{1:\infty}, a_{0:\infty}) \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right]$$

Note that $p(s_{1:\infty}, a_{0:\infty}) = p(s_{1:\infty}, a_{1:\infty}) p(a_0) = p(s_{1:\infty}, a_{1:\infty}) \pi(a_0 | s_0 = s)$. Thus, we obtain the next equality shown below.

$$V^{\pi}(s) = \sum_{s_{1:\infty}, a_{0:\infty}} p(s_{1:\infty}, a_{1:\infty}) \pi(a_0 | s_0 = s) \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s \right]$$

We can then split the summation into two nested summations since $\pi(a_0 | s_0 = s)$ only needs to be summed over all possible a_0, as shown below. Note that by having a separate summation for a_0, we must include the term in the innermost summation as well to obtain marginalization.

$$V^{\pi}(s) = \sum_{a} \pi(a_0 = a | s_0 = s) \sum_{s_{1:\infty}, a_{1:\infty}} p(s_{1:\infty}, a_{1:\infty}) \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right]$$

By applying the definition of expectation $E[X] = \sum_i x_i p(x_i)$, we can collapse the inner summation as shown below.

$$V^{\pi}(s) = \sum_{a} \pi(a_0 = a | s_0 = s) \mathbb{E}_p \left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a \right]$$

Note that the expectation expression is exactly the state-action value function definition $Q^{\pi}(s, a)$, so we can replace the term as shown below.

$$V^{\pi}(s) = \sum_{a} \pi(a_0 = a | s_0 = s) Q^{\pi}(s, a)$$

Thus, we have shown the relationship between the state value function $V^{\pi}(s)$ and state-action value function $Q^{\pi}(s, a)$ is given as $V^{\pi}(s) = \sum_a \pi(a|s) Q^{\pi}(s, a)$. The purpose of the state value function (or state-action value function) is to derive a policy in RL. This is because the state value function (or state-action value function) encodes the future reward, so finding the policy that maximizes

this yields the optimal policy. We can now define a recursive relationship among each state value function known as the Bellman equation as shown below.

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} p(s'|s,a)[r(s',a,s) + \gamma V^{\pi}(s')]$$

Here, $\pi(a|s)$ represents following a fixed policy, $p(s'|s,a)$ is the transition probability, $r(s',a,s)$ is the reward for the respective transition, and $V^{\pi}(s')$ is the 'neighboring' state value function. Thus, the two summations exhaustively consider all possible transitions to neighboring states in the MDP. We now show that this relationship holds. We start with the definition of V^{π} as shown below.

$$V^{\pi}(s_0) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0] = \mathbb{E}_p\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0\right]$$

We can remove r_0 from the summation and change the summation index accordingly as shown below.

$$V^{\pi}(s_0) = \mathbb{E}_p\left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0\right]$$

Next, we apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ and split the summation into two nested summations as shown below.

$$V^{\pi}(s_0) = \sum_{s_{1:\infty}, a_{0:\infty}} p(s_{1:\infty}, a_{0:\infty}) \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0\right] = \sum_{a_{0:\infty}} \sum_{s_{1:\infty}} p(s_{1:\infty}, a_{0:\infty}) \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0\right]$$

Note that $p(s_{1:\infty}, a_{0:\infty}) = p(s_{2:\infty}, a_{1:\infty}) p(s_1, a_0)$ and $p(s_1, a_0) = p(s_1|a_0)p(a_0) = p(s_1|s_0, a_0)\pi(a_0|s_0)$. Using this and by splitting each summation into two nested summations, we obtain the next expressions shown below.

$$V^{\pi}(s_0) = \sum_{a_0} \sum_{s_1} p(s_1, a_0) \left[r_0 + \sum_{a_{1:\infty}} \sum_{s_{2:\infty}} p(s_{2:\infty}, a_{1:\infty}) \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]$$

$$V^{\pi}(s_0) = \sum_{a_0} \sum_{s_1} p(s_1|s_0, a_0)\pi(a_0|s_0) \left[r_0 + \sum_{a_{1:\infty}} \sum_{s_{2:\infty}} p(s_{2:\infty}, a_{1:\infty}) \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]$$

By applying the definition of expectation $E[X] = \sum_i x_i p(x_i)$, we can collapse the inner summation as shown below.

$$V^{\pi}(s_0) = \sum_{a_0} \sum_{s_1} p(s_1|s_0, a_0)\pi(a_0|s_0) \left[r_0 + \mathbb{E}_p\left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1\right]\right]$$

Note that the expectation expression is exactly $\gamma V^{\pi}(s_1)$, so we can replace the term and rearrange the summations as shown below.

$$V^{\pi}(s_0) = \sum_{a_0} \pi(a_0|s_0) \sum_{s_1} p(s_1|s_0, a_0) \left[r_0 + \gamma V^{\pi}(s_1)\right]$$

This holds recursively regardless of the state. The proof of the recursive relationship of neighboring state-action value functions is similar. Once again, we start with the definition of $Q^{\pi}(s_0, a_0)$ as shown below.

$$Q^{\pi}(s_0, a_0) = \mathbb{E}_p[\gamma^0 r_0 + \gamma^1 r_1 + \gamma^2 r_2 + \ldots | s_0, a_0] = \mathbb{E}_p\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0, a_0\right]$$

We can remove r_0 from the summation and change the summation index accordingly as shown below.

$$Q^\pi(s_0, a_0) = \mathbb{E}_p \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0, a_0 \right]$$

Next, we apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ and split the summation into two nested summations as shown below.

$$Q^\pi(s_0, a_0) = \sum_{s_{1:\infty}} p(s_{1:\infty}, a_{1:\infty}) \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0, a_0 \right]$$

Note that $p(s_{1:\infty}, a_{1:\infty}) = p(s_{2:\infty}, a_{1:\infty}) p(s_1)$ and $p(s_1) = p(s_1|s_0, a_0)$. Using this and splitting the summation into two nested sums, we can then apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ to obtain the expressions shown below.

$$Q^\pi(s_0, a_0) = \sum_{s_1} p(s_1|s_0, a_0) \left[r_0 + \sum_{s_{2:\infty}} p(s_{2:\infty}, a_{1:\infty}) \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0, a_0 \right]$$

$$Q^\pi(s_0, a_0) = \sum_{s_1} p(s_1|s_0, a_0) \left[r_0 + \mathbb{E}_p \left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1 \right] \right]$$

We can insert a dependence of the expectation on a_1 and marginalize it to obtain the expression shown below.

$$Q^\pi(s_0, a_0) = \sum_{s_1} p(s_1|s_0, a_0) \left[r_0 + \sum_{a_1} \pi(a_1|s_1) \mathbb{E}_p \left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1, a_1 \right] \right]$$

Note that the expectation expression is exactly $Q^\pi(s_1, a_1)$, so we can replace the term as shown below.

$$Q^\pi(s_0, a_0) = \sum_{s_1} p(s_1|s_0, a_0) \left[r_0 + \sum_{a_1} \pi(a_1|s_1) Q^\pi(s_1, a_1) \right]$$

This holds recursively regardless of the state-action pair. These Bellman equations are important for the following reasons:

- They define a recursive relationship between neighboring value functions
- They are used to derive the Bellman optimality equation
- They provide a constraint for optimization
- They are used in a myriad of algorithms for solving optimal value or policy functions

3.4 Optimal Policies

A policy is said to be better if it provides more reward to the agent. That is, π_1 is strictly better than π_2 if $V^{\pi_1}(s) > V^{\pi_2}(s) \, \forall s$ or $Q^{\pi_1}(s, a) > Q^{\pi_2}(s, a) \, \forall s, a$. Thus, we can state that the best policy π^* ensures that $V^{\pi^*}(s) = \max_\pi V^\pi(s) \, \forall s$ or $Q^{\pi^*}(s, a) = \max_\pi Q^\pi(s, a) \, \forall s, a$. Recall that $V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s, a)$. Under an optimal policy π^*, then it follows that $V^{\pi^*}(s) = \max_a Q^{\pi^*}(s, a)$. This

is because the optimal policy will always choose the action a that maximizes the total reward in state s, so it is inherently deterministic. The Bellman optimality equations, which are recursive relationship between value functions under an optimal policy, are shown below.

$$V^{\pi^*}(s) = \max_a \sum_{s'} p(s'|s,a)[r_t + \gamma V^{\pi^*}(s')]$$

$$Q^{\pi^*}(s,a) = \sum_{s'} p(s'|s,a)[r(s) + \gamma \max_{a'} Q^{\pi^*}(s',a')]$$

There are alternative forms of the above equations that are equivalent. These are shown below.

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a)$$

$$Q^{\pi^*}(s,a) = \sum_{s'} p(s'|s,a)[r(s) + \gamma V^{\pi^*}(s')]$$

We can derive the Bellman optimality equation (for the state value function) by first starting with the relationship between $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ that holds under an optimal policy π^* shown below.

$$V^{\pi^*}(s) = \sum_a \pi^*(a|s) Q^{\pi^*}(s,a)$$

Since $\pi^*(a|s)$ is optimal, it follows that $\pi^*(a|s)$ will equal 1 for the action a that maximizes $Q^{\pi^*}(s,a)$. Thus, we can rewrite the above expression as a maximization as shown below.

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a)$$

We can now replace $Q^{\pi^*}(s,a)$ with the definition $Q^{\pi}(s,a) = \mathbb{E}_p\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$ as shown below.

$$V^{\pi^*}(s) = \max_a \mathbb{E}_p\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$$

We can remove r_0 from the summation and change the summation index accordingly as shown below.

$$V^{\pi^*}(s) = \max_a \mathbb{E}_p\left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right]$$

Next, we apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ and split the summation into two nested summations as shown below.

$$V^{\pi^*}(s) = \max_a \sum_{s_{1:\infty}} p(s_{1:\infty}, a_{1:\infty}) \left[r_0 + \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0 = s, a_0 = a\right]$$

Note that $p(s_{1:\infty}, a_{1:\infty}) = p(s_{2:\infty}, a_{1:\infty}) p(s_1)$ and $p(s_1) = p(s_1|s_0,a_0)$. Let $s_1 = s'$, then $p(s_{1:\infty}, a_{1:\infty}) = p(s_{2:\infty}, a_{1:\infty}) p(s_1 = s')$ and $p(s_1 = s') = p(s_1 = s'|s,a)$. Using this and splitting the summation into two nested sums, we can then apply the definition of expectation $E[X] = \sum_i x_i p(x_i)$ to obtain the expressions shown below.

$$V^{\pi^*}(s_0) = \max_a \sum_{s_1} p(s_1|s_0,a_0) \left[r_0 + \sum_{s_{2:\infty}} p(s_{2:\infty}, a_{1:\infty}) \sum_{t=1}^{\infty} \gamma^t r_t \middle| s_0, a_0\right]$$

$$V^{\pi^*}(s) = \max_a \sum_{s'} p(s_1 = s'|s, a) \left[r_0 + \mathbb{E}_p \left[\sum_{t=1}^{\infty} \gamma^t r_t \middle| s_1 = s' \right] \right]$$

Note that the expectation expression is exactly $\gamma V^{\pi^*}(s_1 = s')$, so we can replace the term and rearrange the summations as shown below.

$$V^{\pi^*}(s) = \max_a \sum_{s'} p(s_1 = s'|s, a) \left[r_0 + \gamma V^{\pi^*}(s_1 = s') \right]$$

Since the 0-th time step is arbitrary, we can obtain the desired expression by abstracting r_0 as r_t to obtain the desired expression as shown below.

$$V^{\pi^*}(s) = \max_a \sum_{s'} p(s'|s, a) \left[r_t + \gamma V^{\pi^*}(s') \right]$$

Note that the maximization operator included in the Bellman optimality equation (as opposed to the standard Bellman equation) is what differentiates the two equations. While the standard Bellman equation describes the expected total reward under any arbitrary policy, the Bellman optimality equation maximizes the expected total reward under a stationary policy. Using the Bellman optimality, we can solve for optimal policies, which is the goal of RL.