ECE598ICM: Interplay between Control and Machine Learning **Fall 2020**

Solutions for Homework 2

1

(a) Denote the policy in the problem statement as μ. We can solve the Bellman equation. The state value function J_μ is just a vector. We can solve J_μ as $J_\mu = (I - \gamma P_\mu)^{-1}\bar{c}_\mu$. Here $\bar{c}_\mu(i) = p_1 c(i, a_1) + (1 - p_1)c(i, a_2)$. The (i, j)-th entry of P_μ is defined as $p_1 P(j, i, a_1) + (1 - p_1)P(j, i, a_2)$ where $P(j, i, a) := P(s_{t+1} = j | s_t = i, a_t = a)$.

We can also solve the Bellman equation for the Q-factor. Specifically, we have

$$Q(i, a) = c(i, a) + \gamma \sum_{j=1}^{n} P(j, i, a)[p_1 Q(j, a_1) + (1 - p_1)Q(j, a_2)]$$

which is equivalent to another linear equation $Q_\mu = \hat{c}_\mu + \gamma M_\mu Q_\mu$ where the i-th row of M_μ is $\begin{bmatrix} p_1 P(1, i, a_1) & (1 - p_1)P(1, i, a_2) & p_1 P(2, i, a_1) & (1 - p_1)P(2, i, a_2) \dots p_1 P(n, i, a_1) & (1 - p_1)P(n, i, a_2) \end{bmatrix}$, and $(Q_\mu, \hat{c}_\mu)$ can be calculated as

$$Q_\mu = \begin{bmatrix} Q(1, a_1) \\ Q(1, a_2) \\ Q(2, a_1) \\ Q(2, a_2) \\ \vdots \\ Q(n, a_1) \\ Q(n, a_2) \end{bmatrix} \quad , \quad \hat{c}_\mu = \begin{bmatrix} c(1, a_1) \\ c(1, a_2) \\ c(2, a_1) \\ c(2, a_2) \\ \vdots \\ c(n, a_1) \\ c(n, a_2) \end{bmatrix}$$

Then Q_μ can be calculated as $Q_\mu = (I - \gamma M_\mu)^{-1}\hat{c}_\mu$. When the transition model is unknown, one can apply temporal difference learning to learn value functions directly.

(b) At every t, the action a_t is generated using the policy μ given in Problem 2(a). Next apply a_t and measure s_{t+1} and $c(s_t, a_t)$. Then update the Q-factor as

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(c(s_t, a_t) + \gamma \max[Q_t(s_{t+1}, a_1), Q_t(s_{t+1}, a_2)] - Q_t(s_t, a_t)\right)$$

The size of the Q-table is $2n$. If $s_t = i$ and $a_t = a_j$, then we only update the $(2i + j - 2)$-th entry of the Q-table at step t.

(c) For SARSA, we need to specify an initial action a_0 (which can be generated arbitrarily). At every step t, apply the action a_t, and measure s_{t+1} and $c(s_t, a_t)$. Use Q_t to generate an ε-greedy policy and then use this policy to sample an action a_{t+1}. Then update the Q-factor as

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \left(c(s_t, a_t) + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)\right)$$

So at step $t \geq 1$, the action a_t is already generated using the ε-greedy policy based on Q_{t-1}.

We can see that Q-learning is off-policy in the sense that the choice of behavior policy can be independent of Q_t. In contrast, SARSA is on policy since the behavior policy is directly related to Q_t. Another difference is that in the update rules, Q-learning requires calculating $\max_{a'} Q_t(s_{t+1}, a')$ (which is equal to $\max[Q_t(s_{t+1}, a_1), Q_t(s_{t+1}, a_2)]$ in this problem) and SARSA directly applies $Q_t(s_{t+1}, a_{t+1})$.

2

(a) Given a linear policy K, it is straightforward to use induction to show

$$V(x) = r_K + \quad x^\mathsf{T} \left(\sum_{t=0}^{\infty} \gamma^t ((A - BK)^\mathsf{T})^t (Q + K^\mathsf{T} R K)(A - BK)^t \right) x \tag{1}$$

where r_K is some extra term introduced by the noise w_t. Therefore, we can parameterize the value function as $x^\mathsf{T} P_K x + r_K$. Therefore, we have

$$V(x) = x^\mathsf{T}(Q + K^\mathsf{T} R K)x + \gamma \left(\mathbb{E}((A - BK)x + w)^\mathsf{T} P_K ((A - BK)x + w) + r_K \right) \tag{2}$$

Notice w is independent from x and has a zero mean, we have

$$\mathbb{E}((A - BK)x + w)^\mathsf{T} P_K ((A - BK)x + w) = x^\mathsf{T}(A - BK)^\mathsf{T} P_K (A - BK)x + \mathbb{E}(w^\mathsf{T} P_K w)$$

Notice that the left side of (2) is just $x^\mathsf{T} P_K x + r_K$. Hence (2) can be rewritten as

$$x^\mathsf{T} P_K x + r_K = x^\mathsf{T}(Q + K^\mathsf{T} R K)x + \gamma x^\mathsf{T}(A - BK)^\mathsf{T} P_K (A - BK)x + \gamma \mathbb{E}(w^\mathsf{T} P_K w) + \gamma r_K$$

To ensure that the quadratic functions on the left and right sides of the above equation are the same, the following have to be true:

$$\begin{aligned} x^\mathsf{T} P_K x &= x^\mathsf{T}(Q + K^\mathsf{T} R K)x + \gamma x^\mathsf{T}(A - BK)^\mathsf{T} P_K (A - BK)x \\ r_K &= \gamma \mathbb{E}(w^\mathsf{T} P_K w) + \gamma r_K \end{aligned}$$

Hence, the Bellman equation becomes

$$P_K = Q + K^\mathsf{T} R K + \gamma (A - BK)^\mathsf{T} P_K (A - BK)$$

and $r_K = \frac{\gamma}{1-\gamma} \mathbb{E}(w^\mathsf{T} P_K w) = \frac{\gamma}{1-\gamma} \operatorname{trace}(PW)$ where W is the covariance matrix of w_t.

For the $\mathcal{Q}$-function, we have

$$\begin{aligned} \mathcal{Q}(x, u) &= x^\top Q x + u^\top R u + \gamma \mathbb{E} V(Ax + Bu + w) \\ &= x^\top Q x + u^\top R u + \gamma \mathbb{E}(Ax + Bu + w)^\mathsf{T} P_K (Ax + Bu + w) + \gamma r_K \\ &= x^\top Q x + u^\top R u + \gamma (Ax + Bu)^\mathsf{T} P_K (Ax + Bu) + \gamma (\mathbb{E}(w^\mathsf{T} P_K w) + r_K) \\ &= \begin{bmatrix} x \\ u \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q + \gamma A^\mathsf{T} P_K A & \gamma A^\mathsf{T} P_K B \\ \gamma B^\mathsf{T} P_K A & R + \gamma B^\mathsf{T} P_K B \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + r_K \end{aligned}$$

We can also directly parameterize $\mathcal{Q}(x,u)$ as

$$\mathcal{Q}(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} \\ \mathcal{Q}_{12}^{\mathsf{T}} & \mathcal{Q}_{22} \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + r_K$$

Notice $V(x) = \mathcal{Q}(x, -Kx)$. Therefore, we can substitute this into the above equation to obtain the Bellman equation for $\mathcal{Q}$:

$$\mathcal{Q}(x,u) = x^\top Q x + u^\top R u + \gamma \mathbb{E}\mathcal{Q}\left(Ax + Bu + w, -K(Ax + Bu + w)\right)$$

which is equivalent to

$$\begin{bmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} \\ \mathcal{Q}_{12}^{\mathsf{T}} & \mathcal{Q}_{22} \end{bmatrix} = \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} + \gamma \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} \\ \mathcal{Q}_{12}^{\mathsf{T}} & \mathcal{Q}_{22} \end{bmatrix} \begin{bmatrix} A & B \\ -KA & -KB \end{bmatrix}$$

$$r_K = \gamma \mathbb{E} \begin{bmatrix} w \\ -Kw \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} \mathcal{Q}_{11} & \mathcal{Q}_{12} \\ \mathcal{Q}_{12}^{\mathsf{T}} & \mathcal{Q}_{22} \end{bmatrix} \begin{bmatrix} w \\ -Kw \end{bmatrix} + \gamma r_K$$

When the model is unknown, we can apply the least square temporal difference (LSTD) learning methods to estimate the value functions from data.

(b) Policy iteration: The PI algorithm iterates as $K^{n+1} = \gamma(\gamma B^{\mathsf{T}} P^n B + R)^{-1} B^{\mathsf{T}} P^n A$ where P_n solves the Bellman equation $\gamma(A - BK^n)^{\mathsf{T}} P^n (A - BK^n) + Q + (K^n)^{\mathsf{T}} R K^n = P^n$. Another option is to evaluate $\mathcal{Q}$ for every step and then design a policy which is the greedy policy for $\mathcal{Q}$. Specifically, at every step n, we first solve the $\mathcal{Q}$ Bellman equation to obtain $(\mathcal{Q}_{11}^n, \mathcal{Q}_{12}^n, \mathcal{Q}_{22}^n)$ (the policy evaluation step), and then update the policy as $K_{n+1} = (\mathcal{Q}_{22}^n)^{-1}(\mathcal{Q}_{12}^n)^{\mathsf{T}}$ (the policy improvement step).

If the model is unknown, we can use LSTD to estimate $\mathcal{Q}$-Factor from data. Suppose we choose the feature as $\phi(x,u)$ and parameterize the $\mathcal{Q}$-function as $\mathcal{Q}_K(x,u) = \theta^\top \phi(x,u)$. Then we need to fit the weight vector θ. We just generate a trajectory of $\{x_t, u_t\}_{t=0}^T$ using $x_{k+1} = Ax_k + Bu_k + w_k$ and $u_k = -Kx_k + v_k$. Here v_k is some noise added for exploration. We fit θ to minimize the target difference error as

$$\theta \approx \left(\sum_{t=0}^{T-1} \phi(x_t,u_t)(\phi(x_t,u_t) - \gamma\phi(x_{t+1}, -Kx_{t+1}))^\top \right)^{-1} \left(\sum_{t=0}^{T-1} c(x_t,u_t)\phi(x_t,u_t) \right)$$

Notice v_k should be large enough to explore the space thoroughly. One can also generate (x_t, u_t) completely randomly for all t. Here u_t can be completely random, and does not need to be generated from policy K. For example, use a uniform distribution over $[-1000, 1000]$ to generate (x_t, u_t). For all t, generate x_t' as $x_t' = Ax_t + Bu_t + w_t$. now estimate θ as

$$\theta \approx \left(\sum_{t=0}^{T} \phi(x_t,u_t)(\phi(x_t,u_t) - \gamma\phi(x_t', -Kx_t'))^\top \right)^{-1} \left(\sum_{t=0}^{T} c(x_t,u_t)\phi(x_t,u_t) \right)$$

(c) Here the optimal state-action value function is the $\mathcal{Q}$-function associated with the optimal policy. $\mathcal{Q}^*$ can be solved from the optimal Bellman equation. Suppose the optimal state value function is $x^\mathsf{T}Px + r$. We have

$$\begin{aligned} x^\mathsf{T}Px + r &= \min_u (x^\mathsf{T}Qx + u^\mathsf{T}Ru + \gamma \mathbb{E}(Ax + Bu + w)^\mathsf{T}P(Ax + Bu + w) + \gamma r) \\ &= \min_u (x^\mathsf{T}Qx + u^\mathsf{T}Ru + \gamma (Ax + Bu)^\mathsf{T}P(Ax + Bu) + \gamma \mathbb{E}w^\mathsf{T}Pw + \gamma r) \end{aligned}$$

Taking gradient of the function on the right side with respect to u leads to

$$u = -\gamma(R + \gamma B^\mathsf{T}PB)^{-1}B^\mathsf{T}PAx.$$

which can be substituted back to get the following optimal Bellman equation:

$$P = Q + \gamma A^\mathsf{T}PA - \gamma^2 A^\mathsf{T}PB(R + \gamma B^\mathsf{T}PB)^{-1}B^\mathsf{T}PA$$

Then the optimal state-action value function can be calculated as

$$\mathcal{Q}^*(x,u) = \begin{bmatrix} x \\ u \end{bmatrix}^\mathsf{T} \begin{bmatrix} Q + \gamma A^\mathsf{T}PA & \gamma A^\mathsf{T}PB \\ \gamma B^\mathsf{T}PA & R + \gamma B^\mathsf{T}PB \end{bmatrix} \begin{bmatrix} x \\ u \end{bmatrix} + \frac{\gamma}{1-\gamma} \operatorname{trace}(PW)$$

The fitted $\mathcal{Q}$-iteration is value-based, i.e. it repeatedly fits the $\mathcal{Q}$-function and does not generate the associated policy during the learning process. In contrast, the approximate policy iteration searches over the policy space and hence is policy-based. At every step, the API method generates a greedy policy from the $\mathcal{Q}$-function associated with the last policy.

3

(a) The policy gradient theorem states that we can estimate the policy gradient as

$$\nabla \mathcal{C}(\theta) = \mathbb{E} \sum_{t=0}^{\infty} [\gamma^t \Psi_t \nabla_\theta \log \pi_\theta(u_t|x_t)] \tag{3}$$

where θ parameterizes the stochastic policy, and Ψ_t can be calculated using one of the following methods:

- Monte Carlo estimation: $\sum_{t'=t}^{\infty} \gamma^{t'-t} c_{t'}$
- Baselined versions of Monte Carlo estimation: $\sum_{t'=t}^{\infty} (\gamma^{t'-t} c_{t'} - b(x_t))$
- State-action value function: $Q^\pi(x_t, u_t)$
- Advantage function: $A^\pi(x_t, u_t)$
- TD residual: $c_t + \gamma V^\pi(x_{t+1}) - V^\pi(x_t)$

(b) We will use the back propagation algorithm. Recall $u_t \sim \mathcal{N}(W^2\sigma(W^1\sigma(W^0x_t)), \tilde{\sigma}I)$. Therefore, we directly have

$$\log \pi_\theta(u_t|x_t) = -\frac{1}{2}\tilde{\sigma}^{-1}\|u_t - W^2\sigma(W^1\sigma(W^0x_t))\|^2 + C \tag{4}$$

where C is some constant. To perform the back propagation algorithm, we define $z^0 = x_t$, $h^0 = W^0z^0$, $z^1 = \sigma(h^0)$, $h^1 = W^1z^1$, $z^2 = \sigma(h^1)$, and $h^2 = W^2z^2$. Suppose the j-th entry of h^l is $h^l(j)$. Then we further define D^l to be a diagonal matrix whose j-th diagonal entry is equal to the derivative $\sigma'(h^l(j))$. Define $e^2 = \tilde{\sigma}^{-1}(u_t - h^2)$, $e^1 = (W^2D^1)^\mathsf{T}e^2$, and $e^0 = (W^1D^0)^\mathsf{T}e^1$. Therefore, we can apply the back propagation algorithm to show

$$\begin{aligned}
\frac{\partial}{\partial W^2}\log \pi_\theta(u_t|x_t) &= e^2(z^2)^\mathsf{T} \\
\frac{\partial}{\partial W^1}\log \pi_\theta(u_t|x_t) &= e^1(z^1)^\mathsf{T} \\
\frac{\partial}{\partial W^0}\log \pi_\theta(u_t|x_t) &= e^0(z^0)^\mathsf{T}
\end{aligned}$$

One can also expand the above formulas. For example, one can expand $\frac{\partial}{\partial W^2}\log \pi_\theta(u_t|x_t)$ as

$$\frac{\partial}{\partial W^2}\log \pi_\theta(u_t|x_t) = e^2(z^2)^\mathsf{T} = \tilde{\sigma}^{-1}(u_t - W^2\sigma(W^1h(W^0x_t)))(\sigma(W^1\sigma(W^0x_t)))^\mathsf{T}$$

One can expand $\frac{\partial}{\partial W^1}\log \pi_\theta(u_t|x_t)$ and $\frac{\partial}{\partial W^0}\log \pi_\theta(u_t|x_t)$ similarly. The details are omitted. See Section 3.1 of the following survey paper for a detailed treatment of backpropogation:

https://arxiv.org/pdf/1912.08957.pdf

(c) If the model is known, we can derive a closed-form gradient formula as follows. We can take the total derivative of both sides of the Bellman equation to get

$$dP_K = d(K^\mathsf{T}RK) + \gamma d\left((A-BK)^\mathsf{T}P_K(A-BK)\right)$$

By the chain rule, we have

$$\begin{aligned}
&dP_K \\
=&dK^\mathsf{T}RK + K^\mathsf{T}RdK + \gamma(A-BK)^\mathsf{T}dP_K(A-BK) - \gamma dK^\mathsf{T}B^\mathsf{T}P_K(A-BK) - \gamma(A-BK)^\mathsf{T}P_KBdK \\
=&dK^\mathsf{T}\left((R+\gamma B^\mathsf{T}P_KB)K - \gamma B^\mathsf{T}P_KA\right) + \left(K^\mathsf{T}(R+\gamma B^\mathsf{T}P_KB) - \gamma A^\mathsf{T}P_KB\right)dK \\
&+ \gamma(A-BK)^\mathsf{T}dP_K(A-BK)
\end{aligned}$$

If we view dP_K as the variable, the above is a Bellman equation which can be solved as

$$dP_K = \sum_{t=0}^{\infty}\gamma^t((A-BK)^\mathsf{T})^t(dK^\mathsf{T}E_K + E_K^\mathsf{T}dK)(A-BK)^t$$

where $E_K = (R+\gamma B^\mathsf{T} P_K B)K - \gamma B^\mathsf{T} P_K A$. By definition, we have $d\mathcal{C}(K) = \sum_{i,j} \frac{\partial \mathcal{C}}{\partial K_{ij}} dK_{ij} = \text{trace}(\nabla \mathcal{C}(K) dK^\mathsf{T})$. Since $\mathcal{C}(K) = \text{trace}(P_K \Sigma_0) + \frac{\gamma}{1-\gamma} \text{trace}(P_K W)$, it is straightforward to show $\nabla \mathcal{C}(K) = 2E_k \Sigma_K$ where $\Sigma_K = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}[x_t x_t^\mathsf{T}]$.

.

4

The key finding is that the LSPI algorithm works efficiently for this problem while it is much more difficult to make the fitted Q-iteration work. This demonstrates that it is relatively easier to make policy-based RL methods work for control problems. A code for the LSPI implementation is provided on the course website. Based on the simulation, the LSPI method can converge to the optimal control gain within 10 iterations. See the code for how to setup behavior policy for efficient exploration.