ECE586RL: MDPs and Reinforcement Learning **Spring 2020**

Homework 1

Instructor: Bin Hu Due date: March 11, 2020

1.Markov Chains. (20 points in total, 10 points for each subproblem)

(a) Consider Markov chains on a finite state space $\mathcal{S} = \{1, 2, 3, 4\}$. Consider the following three different transition matrices.

$$P_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0.2 & 0 & 0.8 \\ 0.1 & 0.2 & 0.7 & 0 \\ 0 & 0 & 0.1 & 0.9 \end{bmatrix}, P_2 = \begin{bmatrix} 0.3 & 0.7 & 0 & 0 \\ 0.4 & 0.6 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0.5 & 0.5 \end{bmatrix}, P_3 = \begin{bmatrix} 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0.4 & 0.6 \\ 0.8 & 0.2 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0 \end{bmatrix}$$

For the above three cases, answer i) Is the chain irreducible? ii) Is the chain aperiodic? (iii) Does the chain have a unique stationary distribution? (iv) Does the chain always converge to its stationary distribution?

(b) Consider a dynamical system $x_{k+1} = Ax_k + w_k$ where w_k is sampled from a Gaussian distribution ($w_k \sim \mathcal{N}(0, W)$) in an IID manner. (i) Suppose x_k has been observed and is known, what is the probability distribution of x_{k+1}? Write out its probability density function. (ii) Given the mean and covariance of x_k, are the mean and covariance of x_{k+1} completely determined? If so, write down the formulas for the mean and covariance of x_{k+1}.

2. MDPs on Finite State Space (20 points in total, 5 points for each subproblem)

Consider a finite state MDP $\langle \mathcal{S}, \mathcal{A}, P, c, \gamma \rangle$. Suppose the state takes value on the set $\{1, 2, \ldots, n\}$ and the action takes value on the set $\{a_1, a_2\}$.

(a) Consider a stochastic policy that takes the action a_1 with probability 0.6 for any state. In other words, the policy selects the action a_2 with probability 0.4 for any state. Suppose the transition matrix is know. How to evaluate the state value function for this policy? How to evaluate the state-action value function for this policy? How about the cases where the transition matrix is not known? Provide one method here.

(b) Suppose we use the policy in (a) as our behavior policy and apply $\mathcal{Q}$-learning to solve the above MDP. What is the update rule here? What is the size for the $\mathcal{Q}$ table?

(c) Now suppose we want to apply SARSA. What is the difference between SARSA and $\mathcal{Q}$-learning?

(d) Suppose we apply the value iteration algorithm to find the optimal $\mathcal{Q}$-function. Your task is to use the contraction mapping theorem to prove the convergence of the VI algorithm. (Hint: First prove that the Bellman operator for $\mathcal{Q}$-factor is a contraction mapping.)

3. Linear Quadratic Regulator (20 points in total, 5 points for each subproblem)
Consider the linear time-invariant system

$$x_{k+1} = Ax_k + Bu_k + w_k$$

where x_k is the state, u_k is the control action, and the process noise w_k is sampled from a Gaussian distribution in an IID manner, i.e. $w_k \sim \mathcal{N}(0, W)$. The objective is to choose u_k to minimize the following cost

$$\mathcal{C} = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}(x_k^\mathsf{T} Q x_k + u_k^\mathsf{T} R u_k)$$

where $0 < \gamma < 1$ is the discounting factor. The matrices Q and R are positive definite.

(a) Policy evaluation: Suppose we are using a linear policy $u_k = -Kx_k$. How to calculate the state value function $\mathcal{C}_K(x) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k (x_k^\mathsf{T} Q x_k + u_k^\mathsf{T} R u_k) \big| x_0 = x\right]$? How to calculate the state-action value function $\mathcal{Q}_K(x,u) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k (x_k^\mathsf{T} Q x_k + u_k^\mathsf{T} R u_k) \big| x_0 = x,\, u_0 = u\right]$? Derive the Bellman equations for both cases.

(b) Optimal Bellman equation: We define the optimal state value function as $\mathcal{C}^*(x) = \min_\pi \mathcal{C}_\pi(x)$. How to calculate $\mathcal{C}^*$? What is the optimal state-action value function $\mathcal{Q}^*$ here?

(c) Approximate Policy Iteration: Write out the policy iteration algorithm for the above problem. In the policy evaluation step, how to estimate $\mathcal{Q}_K$ from sample trajectories of $\{x_k, u_k\}$? Provide at least two algorithms for that. Write out the specific update rules for the algorithms you provide.

(d) $\mathcal{Q}$-learning and SARSA: write out the update rules for $\mathcal{Q}$-learning and SARSA when applied to the above LQR problem. What is the main difference?

4. Policy Gradient
Consider a nonlinear system $x_{k+1} = f(x_k, u_k, w_k)$ where x_k is the state, u_k is the action, and w_k is the process noised sampled from an IID Gaussian distribution. The objective is to choose u_k to minimize the cost

$$\mathcal{C} = \mathbb{E} \sum_{k=0}^{\infty} \gamma^k c(x_k, u_k) \tag{1}$$

where $0 < \gamma < 1$ is the discounting factor. Suppose f is unknown, and we want to learn policy from sampled trajectories of (x_k, u_k).

(a) Based on the policy gradient theorem, we can estimate the policy gradient as

$$\nabla \mathcal{C}(\theta) = \mathbb{E}\sum_{t=0}^{\infty}[\Psi_t \nabla_\theta \log \pi_\theta(u_t|x_t)] \tag{2}$$

where θ parameterizes the stochastic policy. Many options for Ψ_t are available. Provide three such options. If we use a linear Gaussian policy, i.e. $u_t \sim \mathcal{N}(-Kx, \sigma I)$, how to estimate the policy gradient? (Write out an explicit formula for $\log \pi_\theta(u_t|x_t)$ and substitute it into the gradient formula.) If we use a two-layer neural network to parameterize the Gaussian policy, i.e. $u_t \sim \mathcal{N}(W^1 h(W^0 x_t), \sigma I)$ where h is the elementwise activation, how to estimate the policy gradient?

(b) When using the policy gradient theorem, we inject noise into the control actions for exploration purposes. Suppose now we want to directly learn a deterministic policy using evolution strategies which are based on the following estimation of the policy gradient:

$$\nabla \mathcal{C}(K) \approx \frac{\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\sigma^2 I)}\mathcal{C}(K+\varepsilon)\varepsilon}{\sigma^2}$$

To understand this update rule, we analyze a shifted variant of the above update:

$$g = \frac{\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,\sigma^2 I)}(\mathcal{C}(K+\varepsilon)-\mathcal{C}(K))\varepsilon}{\sigma^2} = \frac{\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}(\mathcal{C}(K+\sigma\varepsilon)-\mathcal{C}(K))\varepsilon}{\sigma}$$

Roughly speaking, the above estimation shifts the original zeroth-order gradient estimate with a zero mean vector and should not change the mean of the gradient estimator. Your task is to apply the fact $\lim_{\sigma\to 0}\frac{\mathcal{C}(K+\sigma\varepsilon)-\mathcal{C}(K)}{\sigma} = (\nabla\mathcal{C}(K))^\mathsf{T}\varepsilon$ to show the following equation:

$$\mathbb{E}_{\varepsilon\sim\mathcal{N}(0,I)}\left(\lim_{\sigma\to 0}\frac{\mathcal{C}(K+\sigma\varepsilon)-\mathcal{C}(K)}{\sigma}\right)\varepsilon = \nabla\mathcal{C}(K)$$

(Remark: Now you can see σ serves as a stepsize for the stochastic finite difference estimation. In the setting of data-driven control, typically we only have samples of $\mathcal{C}$. Choosing σ to be too small can amplify this error, and hence one has to tune σ carefully.)

5. **Implementation Assignment** (20 points in total, 10 points for each subproblem)
In this problem, you are asked to implement codes for a LQR example with the following parameters

$$A = \begin{bmatrix} 0.99 & 0.01 & 0 \\ 0.01 & 0.98 & 0.01 \\ 0.5 & 0.12 & 0.97 \end{bmatrix}, B = \begin{bmatrix} 1 & 0.1 \\ 0 & 0.1 \\ 0 & 0.1 \end{bmatrix}, Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, W = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.05 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}.$$

Here W is the covariance matrix of the process noise which is sampled from zero mean Gaussian process. We fix the discounting factor as $\gamma = 0.98$.

(a) Model-based policy evaluation: Consider a policy $u_k = -Kx_k$ with $K = \begin{bmatrix} 0 & 0.1 & 0 \\ 0.1 & 0 & 0.1 \end{bmatrix}$. First, calculate the state-action value function using the information of (A, B, Q, R, W). You are allowed to use the matlab function `dlyap`. Or you can use any language to implement a code that solves the LQR Bellman equation for the $\mathcal{Q}$ factor.

(b) Model-free policy evaluation: Still consider the same policy. Now suppose you don't know (A, B, W). You can only access the sample trajectories. Implement TD learning (pick your favorite algorithm from TD(0), TD(λ), or LSTD) to estimate the $\mathcal{Q}$ function. Does the algorithm work? Discuss your finding.

(c) Bonus problem (extra 5 points): Implement the approximate PI algorithm to obtain the optimal policy for the above LQR problem. You can initialize the algorithm using the policy in (a). For the policy evaluation step, choose your favorite TD algorithm.