Homework Set 4

Introduction to Artificial Intelligence with Mathematics (MAS473)

Total Points = 50pts

1. (15pts) In this problem, we consider a toy example of applying a Markov Chain Monte Carlo (MCMC) method. Let X be a discrete random variable whose unnormalized probability mass function is

$$\tilde{p}(X=1)=4, \tilde{p}(X=2)=2, \tilde{p}(X=3)=3, \tilde{p}(X=4)=3.$$

Our goal is to evaluate $\mathbf{E}[f(X)]$ where $f(x)=x^2-3$. To apply the Metropolis-Hasting algorithm, define two proposal distributions $q(i,j)(=q_{ij})$ and $\tilde{q}(i,j)(=\tilde{q}_{ij})$ where

$$q(i,i+1)=q(i,i-1)=\frac{1}{2}, \quad \tilde{q}(i,i+1)=\frac{2}{3}, \tilde{q}(i,i-1)=\frac{1}{3}.$$

(If $i+1=5$, consider 5 as 1. Also, if $i-1=0$, consider 0 as 4.)

   (a) Using the Metropolis-Hasting algorithm, construct Markov chains for proposal distributions q and $\tilde{q}$ when we set the initial state $X_0=1$. You should find the transition matrices for these Markov chains.

   (b) For q and $\tilde{q}$, make a python code to evaluate the error between the actual value of $\mathbf{E}[f(X)]$ and $\mathbf{E}\left[\frac{1}{n}\sum_{i=0}^{n-1} f(X_i)\right]$ where $\{X_i\}$ is the Markov chain from (a) and plot the errors along $n=1,\cdots,1000$. Which proposal distribution converges faster in the expected sense? You should calculate the exact $\mathbf{E}[f(X)]$ and $\mathbf{E}\left[\frac{1}{n}\sum_{i=0}^{n-1} f(X_i)\right]$, not a sample mean of $f(X)$ and $\frac{1}{n}\sum_{i=0}^{n-1} f(X_i)$.

2. (15pts) Consider a Markov decision process (S,A,P,R) where $S=\{0,1,2,3,4\}$ is a state space, $A=\{a^1,a^2\}$ is an action space, P is a transition probability matrix such that

$$P((s+1)\%5|s,a^1)=P((s+2)\%5|s,a^1)=P((s+4)\%5|s,a^1)=\frac{1}{3},$$
$$P((s+1)\%5|s,a^2)=P((s+3)\%5|s,a^2)=\frac{1}{2}$$

(where $a\%b$ means the remainder when a is divided by b) for all $s\in S$ and R is the reward such that $R(s,a,s')$ follows Bernoulli($\frac{1}{2}$) if $s'=0$ and 0 otherwise.[1] Also, when the process reaches the state 4, then the process is terminated (i.e. 4 is the terminal state). Assume the initial state is $s_0=0$ and set the discounted factor $\gamma=0.9$. Let π be a Markovian randomized stationary policy that $\pi(a^1|s)=\pi(a^2|s)=0.5$ for $s=0,2$ and $\pi(a^1|s)=0.7, \pi(a^2|s)=0.3$ for $s=1,3$. In this problem, you may use the python code for matrix calculations, e.g. matrix addition, multiplication, inversion, ...

   (a) When we adopt the policy π, find the probability that the following trajectory is sampled:

$$\tau=\big(s_0=0,\ a_0=a^1,\ r_0=0,\ s_1=2,\ a_1=a^2,\ r_1=1,\ s_2=0,$$
$$a_2=a^2,\ r_2=0,\ s_3=3,\ a_3=a^2,\ r_3=0,\ s_4=4\big)$$

   (b) Calculate $V^\pi(s)$ and $Q^\pi(s,a)$.

[1] In the regular lecture, we use $R(s,a)$ but we can consider a general reward $R(s,a,s')$. It means that the reward when the state is s, the action a is taken and s' is the next state.

(c) Let d_0 be a Markovian deterministic stationary policy that $d_0(s) = a^1$ for all $s \in S$. Using the policy iteration algorithm three times with the initial policy d_0, show your result, e.g. policy evalutation result $v^{(n)}$ and the improved policy d_{n+1}.

(d) Find all Markovian deterministic optimal policies and the optimal value function $V^*(s)$.

3. (5pts) In this problem, we solve a problem to find the shortest path from the Start to the End in the maze using the Markov decision process.

Start (1, 5)	(2, 5)			(5, 5)
(1, 4)	(2, 4)	(3, 4)	(4, 4)	(5, 4)
(1, 3)			(4, 3)	
(1, 2)	(2, 2)		(4, 2)	(5, 2)
	(2, 1)		(4, 1)	End (5, 1)

We formulate this problem as the following:

- The state space $\mathcal{S}$ consists of the white blocks in the figure. Each element is expressed as (i, j).
- The action space $A = \{N, S, W, E\}$ where N, S, W and E mean that go to north, south, west and east, respectively. However, the next state does not change if the direction of action is blocked. For example,

$$P((1,3)|(1,4), S) = 1, \quad P((4,4)|(3,4), E) = 1,$$
$$P((1,5)|(1,5), N) = 1, \quad P((4,3)|(4,4), S) = 1.$$

- $(1, 5)$ is the initial state and $(5, 1)$ is the terminal state.[2]
- $R(s, a, s') = 1$ if $s' = (5, 1)$ and 0 otherwise.
- The discounted factor $\gamma = 0.9$.

Initialize $V_0(s) = 0$ for all $s \in \mathcal{S}$. Using the Bellman optimal operator

$$(Lv)(s) = \sup_{a \in A} \mathbf{E}_{s' \sim P(\cdot|s,a)} \left[R(s, a, s') + \gamma \cdot v(s') \right],$$

set

$$V_{n+1}(s) = (LV_n)(s), \quad n = 0, 1, 2, \cdots.$$

A policy π is said to be a greedy policy induced by a value function V if

$$\pi(s) \in \text{argmax}_{a \in A} Q(s, a)$$

where $Q(s, a) = \mathbf{E}_{s'} [R(s, a, s') + \gamma V(s')]$. Find the minimum $n \in \mathbb{N}$ such that all of greedy policies induced by V_n is an optimal policy. Note that π^* is said to be an optimal policy if $V^{\pi^*}(s) = V^*(s)$ for any $s \in \mathcal{S}$ where V^* is an optimal value function.

[2] The meaning of the terminal state and $R(s, a, s')$ are explained in Problem 2.

4. (15pts) Let $f_\Theta : \mathbb{R}^2 \to \mathbb{R}^2$ be a neural network such that

$$f_\Theta(\mathbf{x}) = W^{(2)}\sigma\left(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) + \mathbf{b}^{(2)}$$

where

$$W^{(1)} \in \mathbb{R}^{3\times 2}, W^{(2)} \in \mathbb{R}^{2\times 3}, \mathbf{b}^{(1)} \in \mathbb{R}^3, \mathbf{b}^{(2)} \in \mathbb{R}^2$$

and σ is the ReLU function. Suppose the parameter $\Theta = \{W^{(1)}, W^{(2)}, \mathbf{b}^{(1)}, \mathbf{b}^{(2)}\}$ is initialized as

$$W^{(1)} = \begin{bmatrix} 1 & 0 & -1 \\ -1 & -1 & 1 \end{bmatrix}^\top, W^{(2)} = \begin{bmatrix} 0 & -2 & 1 \\ 1 & -1 & -1 \end{bmatrix}, \mathbf{b}^{(1)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top, \mathbf{b}^{(2)} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top.$$

To minimize the L^2 loss $\ell(\Theta) = \frac{1}{2}\left\|\mathbf{y} - f_\Theta(\mathbf{x})\right\|^2$, we will use the gradient descent method with a learning rate $\gamma = 1$. Calculate Θ for two iterations of the optimization when we have a training data

$$\mathcal{D} = \{(\begin{bmatrix} 2 & -3 \end{bmatrix}^\top, \begin{bmatrix} -4 & 0 \end{bmatrix}^\top)\}.$$

To solve this problem, you cannot use the programming. Solve it by your hands.