UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

CS440/ECE448 Artificial Intelligence

Practice Exam 3 Spring 2024

The exam will be May 7, 2024

Instructions

- You will be assigned an exam section: May 7, 8:00am or 1:30pm, DCL 1320 or 3039 CIF.
- The exam will contain sixteen multiple-choice questions. Four will be about reinforcement learning. Six will be exam or daily quiz questions repeated from exam 1 or the first third of the course. Six others will be exam or daily quiz questions repeated from exam 2 or the second third of the course.
- This practice exam includes only questions about reinforcement learning. Practice questions about other topics have been released in the practice exams for exam 1 and exam 2.
- No book, notes, or calculator will be allowed.
- The following formula pages, or something very similar, will be available to you attached to the online exam.

Possibly Useful Formulas

Probability:

$$P(X=x) = \Pr(X=x) \text{ or } P(X=x) = \frac{d}{dx}\Pr(X \le x)$$

$$P(X,Y) = P(X|Y)P(Y)$$

$$E[f(X,Y)] = \sum_{x,y} f(x,y)P(X=x,Y=y)$$

MAP Decision: $f(x) = \underset{y}{\operatorname{argmax}} P(Y=y|X=x)$

Bayes Error Rate: $= \sum_x P(X=x) \min_y P(Y \neq y|X=x)$

Precision $= P(Y=1|f(X)=1) = \dfrac{TP}{TP+FP}$

Recall=Sensitivity $= P(f(X)=1|Y=1) = \dfrac{TP}{TP+FN}$

Specificity $= P(f(X)=0|Y=0) = \dfrac{TN}{TN+FP}$

Naive Bayes: $f(x) \approx \underset{y}{\operatorname{argmax}} \left(\ln P(Y=y) + \sum_{i=1}^{n} \ln P(W=w_i|Y=y) \right)$

Laplace Smoothing: $P(W=w_i|Y=y) = \dfrac{k+\text{Count}(w_i,y)}{k+\sum_{v \in \mathscr{V}}(k+\text{Count}(v,y))}$

HMM: $v_1(j) = \pi(j)b_j(\mathbf{x}_t)$

$$v_t(j) = \max_i v_{t-1}(i)a_{i,j}b_j(\mathbf{x}_t), \quad \psi_t(j) = \underset{i}{\operatorname{argmax}}\, v_{t-1}(i)a_{i,j}b_j(\mathbf{x}_t)$$

$$y^*(T) = \underset{i}{\operatorname{argmax}}\, v_T(i), \quad y^*(t) = \psi_{t+1}(y^*(t+1))$$

Demographic Parity: $P(f(X)|A=1) = P(f(X)|A=0)$

Equal Odds: $P(f(X)|Y,A=1) = P(f(X)|Y,A=0)$

Predictive Parity: $P(Y|f(X),A=1) = P(Y|f(X),A=0)$

Learning: $\mathscr{R} = E[\ell(Y,f(X))], \quad \mathscr{R}_{emp} = \dfrac{1}{n}\sum_{i=1}^{n} \ell(y_i,f(x_i))$

Linear Regression : $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$

$$\mathscr{L} = \frac{1}{n}\sum_{i=1}^{n}\mathscr{L}_i, \quad \mathscr{L}_i = \frac{1}{2}\varepsilon_i^2, \varepsilon_i = f(\mathbf{x}_i) - y_i$$

Linear Classifier: $f(\mathbf{x}) = \text{argmax}\,(\mathbf{W}\mathbf{x}+\mathbf{b})$

Perceptron: $\mathbf{w}_c \leftarrow \begin{cases} \mathbf{w}_c - \eta\mathbf{x} & c = \text{argmax}\,(\mathbf{W}\mathbf{x}+\mathbf{b}) \\ \mathbf{w}_c + \eta\mathbf{x} & c = y \\ \mathbf{w}_c & \text{otherwise} \end{cases}$

Softmax: $f_c(\mathbf{x}) = \dfrac{\exp(\mathbf{w}_c^T\mathbf{x}+b_c)}{\sum_{k=1}^{|\mathscr{Y}|}\exp(\mathbf{w}_k^T\mathbf{x}+b_k)} \approx P(Y=c|\mathbf{x})$

Sigmoid: $\sigma(\mathbf{w}^T\mathbf{x}+b) = \dfrac{1}{1+e^{-(\mathbf{w}^T\mathbf{x}+b)}} \approx P(Y=1|\mathbf{x})$

Cross Entropy: $\mathscr{L} = -\ln f_y(\mathbf{x}),\ \dfrac{\partial\mathscr{L}}{\partial f_c(\mathbf{x})} = \begin{cases} -\frac{1}{f_c(\mathbf{x})} & c = y \\ 0 & \text{otherwise} \end{cases}$

SGD: $\mathbf{w}_C \leftarrow \mathbf{w}_c - \eta\dfrac{\partial\mathscr{L}}{\partial\mathbf{w}_c} = \begin{cases} \mathbf{w}_c - \eta(f_c(\mathbf{x}_i)-1)\mathbf{x}_i & c = y \\ \mathbf{w}_c - \eta(f_c(\mathbf{x}_i)-0)\mathbf{x}_i & \text{otherwise} \end{cases}$

Pinhole Camera: $\dfrac{x'}{f} = -\dfrac{x}{z},\ \dfrac{y'}{f} = -\dfrac{y}{z}$

Image Gradient:

$$h_x(x',y') = \frac{h(x'+1,y') - h(x'-1,y')}{2},\ h_y(x',y') = \frac{h(x',y'+1) - h(x',y'-1)}{2}$$

Convolution:

$$y[k,l] = \sum_i\sum_j x[k-i,l-j]h[i,j],\ \frac{\partial y[k,l]}{\partial h[i,j]} = x[k-i,l-j]$$

Max Pooling:

$$z[m] = \max_{(m-1)p+1\le k\le mp} y[k],\ \frac{\partial z[m]}{\partial y[j]} = \begin{cases} 1 & j = \text{argmax}_{(m-1)p+1\le k\le mp}\, y[k] \\ 0 & \text{otherwise} \end{cases}$$

Admissible: $\hat{h}(n) \le h(n)$

Consistent: $\hat{h}(n) - \hat{h}(m) \le h(n,m)$

Value Iteration: $u_i(s) = r(s) + \gamma \max_a \sum_{s'} P(s'|s,a) u_{i-1}(s)$

Policy Evaluation: $u_\pi(s) = r(s) + \gamma \sum_{s'} P(s'|s,\pi(s)) u_\pi(s')$

Policy Improvement: $\pi_{i+1}(s) = \operatorname{argmax}_a \sum_{s'} P(s'|s,a) u_{\pi_i}(s')$

Alpha-Beta Max Node: $v = \max(v, \text{child}); \quad \alpha = \max(\alpha, \text{child})$

Alpha-Beta Min Node: $v = \min(v, \text{child}); \quad \beta = \min(\beta, \text{child})$

Expectiminimax: $u(s) = \begin{cases} \max_a \sum_{s'} P(s'|a,a) u(s') & s \in \text{max states} \\ \min_a \sum_{s'} P(s'|a,a) u(s') & s \in \text{min states} \end{cases}$

Mixed Nash Equilibrium:
$$P(A=0) r_B(0,0) + P(A=1) r_B(1,0) = P(A=0) r_B(0,1) + P(A=1) r_B(1,1)$$
$$P(B=0) r_A(0,0) + P(B=1) r_A(0,1) = P(B=0) r_A(1,0) + P(B=1) r_A(1,1)$$

Unification: $S : \{\mathscr{V}_P, \mathscr{V}_Q\} \to \{\mathscr{V}_Q, \mathscr{C}\}$ such that $S(P) = S(Q) = U$

CBOW Generative: $\mathscr{L} = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=-c, j\neq 0}^{c} \ln \frac{\exp(\mathbf{v}_t^T \mathbf{v}_{t_j})}{\sum_{\mathbf{v} \in \mathscr{V}} \exp(\mathbf{v}^T \mathbf{v}_{t+j})}$

Skip-gram Contrastive: $\mathscr{L} = -\frac{1}{T} \sum_{t=1}^{T} \left(\sum_{\mathbf{v}' \in \mathscr{D}_+(w_t)} \ln \frac{1}{1+e^{-\mathbf{v}'^T \mathbf{v}_t}} + \sum_{\mathbf{v}' \in \mathscr{D}_-(w_t)} \ln \frac{1}{1+e^{\mathbf{v}'^T \mathbf{v}_t}} \right)$

Transformer: $\mathbf{c}_t = \sum_s \alpha(t,s) \mathbf{v}_s$

Attention: $\alpha(t,s) = \frac{\exp(\mathbf{q}_t^T \mathbf{k}_s)}{\sum_{s'} \exp(\mathbf{q}_t^T \mathbf{k}_{s'})}$

Model-based Learning: $P(s_{t+1}|s_t,a_t) = \frac{\text{Count}(s_t,a_t,s_{t+1}) + k}{\sum_{s' \in \mathscr{S}} (\text{Count}(s_t,a_t,s') + k)}$

On-policy learning: $\mathbf{W}_{a_t} \leftarrow \mathbf{W}_{a_t} + \eta \nabla_{\mathbf{W}_{a_t}} \ln P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$

Off-policy learning: $\mathbf{W} \leftarrow \mathbf{W} + \eta \nabla_{\mathbf{W}} \ln P(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)$

Epsilon-first: $N_{first} = \frac{1}{\varepsilon}$

Epsilon-greedy: If $z \leq \varepsilon$ then explore, $z \in (0,1)$

TD-learning: $q_{local}(s_t,a_t) = r_t + \gamma \max_{a' \in \mathscr{A}} q_t(s_{t+1}, a')$

SARSA: $q_{local}(s_t,a_t) = r_t + \gamma q_t(s_{t+1}, at+1)$

Q-learning: $q_{t+1}(s_t,a_t) = q_t(s_t,a_t) + \eta (q_{local}(s_t,a_t) - q_t(s_t,a_t))$

Deep Q: $\theta_{t+1} = \theta_t - \eta \frac{\partial}{\partial \theta} \frac{1}{2} (q_t(s_t,a_t) - q_{local}(s_t,a_t))^2$

Policy Gradient: $\frac{\partial u(s_t)}{\partial \theta} = \sum_\tau \frac{\partial P(\tau)}{\partial \theta} v(\tau) = \sum_\tau P(\tau) \frac{\partial \ln P(\tau)}{\partial \theta} v(\tau) = E\left[\frac{\partial \ln P(\tau)}{\partial \theta} v(\tau) \right]$

Standard error: $M = \frac{1}{n} \sum_{i=1}^{n} Y_i, \quad \text{stdev}(M) = \frac{\sigma}{\sqrt{n}}$

Question 1 ***(0 points)***

In a Markov Decision Process with finite state and action sets, model-based reinforcement learning needs to learn a larger number of trainable parameters than model-free reinforcement learning.

√ **True**

◯ False

Explain:

Solution: Model-based learning needs to learn $P(s'|s,a)$, a set of $N_S^2 N_a$ parameters, where N_s is the number of states, N_a the number of actions. Model-free learning needs to learn $q(s,a)$, a set of only $N_s N_a$ trainable parameters.

Question 2 ***(0 points)***

When we apply the Q-learning algorithm to learn the state-action value function, one big problem in practice may be that the state space of the problem is continuous and high-dimensional. Discuss at least two possible methods to address this.

Solution:

1. Discretize the state space.
2. Design a lower-dimensional set of discrete features to represent the states.
3. Use a parametric approximator (e.g., a neural network) to estimate the Q function values and learn the parameters instead of directly learning the state-action value functions.

Question 3 ***(0 points)***

What is the optimal policy defined by the Bellman equation?

Solution:

$$\pi^*(s) = \operatorname*{argmax}_a \sum_{s'} P(s'|s,a)U(s')$$

Question 4 ***(0 points)***

A cat lives in a two-room apartment. At each point in time, the cat must decide whether to pounce on something, or sleep. The cat decides to implement deep-Q learning so that it can take advantage of all of the information it knows: its new state variable is a vector of six measurements including the (x,y) position of its human servant, the amount of food in its bowl, the temperature of the warm spot in front of the window, and its own (x,y) position in the apartment. Given this state vector, the cat's neural network computes a vector of hidden nodes $\mathbf{h}(s) = [h_1(s), \ldots, h_n(s)]^T$, then computes two outputs corresponding to the two possible actions, $a \in \{\text{pounce}, \text{sleep}\}$. The two outputs of the neural network are

$$q_t(s,a) = \sum_{i=1}^{n} w_{t,a,i} h_i(s), \quad a \in \{\text{pounce}, \text{sleep}\},$$

where $w_{t,a,i}$ are weights that are trained using stochastic gradient descent:

$$w_{t+1,a,i} = w_{t,a,i} - \eta \frac{\partial \mathscr{L}_t}{\partial w_{t,a,i}} \tag{1}$$

Suppose the cat measures the state vector s_t, receives a reward of $r_t = 2$, decides on the action $a_t =$ pounce, and then measures the resulting state vector s_{t+1} at time $t+1$. On the basis of these measurements, propose a loss function $\mathscr{L}_t$ whose derivative could be used to update the neural network as shown in Eq. 2. Specify exactly the way in which your loss function depends on the discount factor γ and on the neural network outputs that might be computed from input vectors s_t ($q_t(s_t,0)$ and/or $q_t(s_t,1)$) and/or from input vector s_{t+1} ($q_t(s_{t+1},0)$ and/or $q_t(s_{t+1},1)$).

Solution: The action a_{t+1} is not specified, so we can't use SARSA, so let's use TD-learning:

$$\begin{aligned} q_{\text{local}}(s_t, a_t = \text{pounce}) &= r(s_t) + \gamma \max_{a'} q_t(s_{t+1}, a') \\ &= 2 + \gamma \max_{a'} q_t(s_{t+1}, a') \end{aligned}$$

A useful loss function might be the squared distance between $q_t(s_t, \text{pounce})$ and $q_{\text{local}}(s_t, \text{pounce})$, which is

$$\mathscr{L} = \frac{1}{2} \left(q_t(s_t, \text{pounce}) - 2 - \gamma \max_{a'} q_t(s_{t+1}, a') \right)^2$$

Question 5 ***(0 points)***

Gepetto the toymaker must decide how many toys to make tomorrow. Gepetto decides to implement deep-Q learning so that he can take advantage of all of the information available to him: his new state variable is a vector of six measurements including the prices of the last three toys sold, two measures of overall economic health, and the number of toys available for sale in his shop. Given this state vector, his neural network computes a vector of hidden nodes $\mathbf{h}(s) = [h_1(s), \ldots, h_n(s)]$, then computes three outputs corresponding to the three possible actions, $a \in \{0,1,2\}$. The three outputs of his neural network are

$$q_t(s,a) = \sum_{i=1}^{n} w_{t,a,i} h_i(s), \quad a \in \{0,1,2\},$$

where $w_{t,a,i}$ are weights that are trained using stochastic gradient descent:

$$w_{t+1,a,i} = w_{t,a,i} - \alpha \frac{\partial \mathscr{L}_t}{\partial w_{t,a,i}} \tag{2}$$

Suppose Gepetto measures the state vector s_t, receives a reward of $r_t = 200$, decides on the action $a_t = 2$, and then measures the resulting state vector s_{t+1} on the following day. On the basis of these measurements, propose a loss function $\mathscr{L}_t$ whose derivative could be used to update the neural network as shown in Eq. 2. Specify exactly the way in which your loss function depends on the discount factor γ and on the neural network outputs that might be computed from input vectors s_t ($q_t(s_t,0), q_t(s_t,1)$, and/or $q_t(s_t,2)$) and/or from input vector s_{t+1} ($q_t(s_{t+1},0), q_t(s_{t+1},1)$, and/or $q_t(s_{t+1},2)$).

Solution: The action a_{t+1} is not specified, so we can't use SARSA, so let's use TD-learning:

$$q_{\text{local}}(s_t, a_t = 2) = r(s_t) + \gamma \max_{a'} q_t(s_{t+1}, a') \qquad = 200 + \gamma \max_{a'} q_t(s_{t+1}, a')$$

A useful loss function might be the squared distance between $q_t(s_t,2)$ and $q_{\text{local}}(s_t,2)$, which is

$$\mathscr{L} = \frac{1}{2}\left(q_t(s_t,2) - 200 - \gamma \max_{a'} q_t(s_{t+1}, a')\right)^2$$

Question 6 ***(0 points)***

Consider an MDP with four states, $s \in \{0,1,2,3\}$. The rewards for these states are $r(0) = 0, r(1) = 1, r(2) = -10000, r(3) = 10000$. From any of these states, there are three possible actions, $a \in \{0,1,2\}$.

Consider a model-based reinforcement learning (RL) algorithm that is trying to learn the transition probabilities for this MDP. Unknown to the RL, the true transition probabilities are the same for every state, and they are:

	$P(s'\|s,a=0)$				$P(s'\|s,a=1)$				$P(s'\|s,a=2)$			
s'	0	1	2	3	0	1	2	3	0	1	2	3
$s=0$	0.5	0.5	0	0	0.99	0	0	0.01	0	0.99	0.01	0
$s=1$	0.5	0.5	0	0	0.99	0	0	0.01	0	0.99	0.01	0
$s=2$	0.5	0.5	0	0	0.99	0	0	0.01	0	0.99	0.01	0

$$P(s'|s,a=0) = \begin{cases} 0.5 & s'=0, \forall s \\ 0.5 & s'=1, \forall s \end{cases}$$

$$P(s'|s,a=1) = \begin{cases} 0.99 & s'=0, \forall s \\ 0.01 & s'=3, \forall s \end{cases}$$

$$P(s'|s,a=2) = \begin{cases} 0.99 & s'=1, \forall s \\ 0.01 & s'=2, \forall s \end{cases}$$

Suppose that a model-based RL algorithm uses an epsilon-first exploration vs. exploitation policy to try to learn these transition probabilities, using $N = \frac{1}{\varepsilon} = 5$ trials per state/action pair. Consider the probable outcome of this experiment. When the exploration is over and exploitation begins, which action is most likely to be chosen as the optimal action? Will the RL continue to view this action as optimal, even after thousands of trials? Is there any action that will probably never be chosen after the initial exploration phase?

Solution:

- When exploitation begins, action $a = 2$ will probably be considered optimal, because 99% of the time this action leads to state 1, which has a positive reward. There is a 1% chance that it might lead to an undesirable outcome, but the probability of this occurring during the five exploratory trials is small.
- After thousands of trials, the RL will have enough observations of action $a = 2$ to see that it leads to a huge loss $(-10,000)$ about once every hundred trials. Therefore, the RL will switch over to considering action $a = 0$ the optimal action, because action $a = 0$ leads to the positive-reward state 50% of the time, and never leads to the negative reward state.
- The RL will probably never use action $a = 1$ after the initial exploration phase. Action $a = 1$ has a high expected payoff because it leads to the high-payoff state with 1% probability, but this event would probably not be observed during the exploration period. If this is not discovered during the initial exploration phase, then the algorithm will never choose action $a = 1$ during the exploitation phase, so it will never learn this fact about action $a = 1$.

Question 7 ***(0 points)***

Consider two different exploration vs. exploitation tradeoff strategies: epsilon-first, and epsilon-greedy. Now consider a hybrid strategy in which you explore every (state,action) pair $N = \frac{1}{\varepsilon}$ times first, then exploit it $100(1-\varepsilon)\%$ of the trials thereafter.

1. Find an advantage that epsilon-first and the hybrid strategy have, relative to epsilon-greedy.
2. Find an advantage that epsilon-greedy and the hybrid strategy have, relative to epsilon-first.
3. Find an advantage that either epsilon-first or epsilon-greedy have, relative to both of the other two strategies.

Solution:

1. Both epsilon-first and the hybrid strategy learn a lot about the environment relatively quickly.
2. Both epsilon-greedy and the hybrid strategy will eventually have an infinite number of trials for every (state,action) pair, so both strategies will eventually converge to perfect knowledge of the environment.
3. Epsilon-first has the advantage that, after the first $N = 1/\varepsilon$ trials of every (state,action) pair, the rest of the trials are spent exploiting the available knowledge, i.e., maximizing reward. Both epsilon-greedy and the hybrid strategy spend $100\varepsilon\%$ of their trials exploring, during which time they are not seeking to maximize reward.

Question 8 ***(10 points)***

A cat lives in a two-room apartment. It has two possible actions: purr, or walk. It starts in room $s_0 = 1$, where it receives the reward $r_0 = 2$ (petting). It then implements the following sequence of actions: a_0 =walk, a_1 =purr. In response, it observes the following sequence of states and rewards: $s_1 = 2$, $r_1 = 5$ (food), $s_2 = 2$.

(a) (3 points) The cat starts out with a Q-table whose entries are all $q(s,a) = 0$, then performs one iteration of TD-learning using each of the two SARS sequences described above (one iteration/time step, for two time steps). Because the cat doesn't like to worry about the distant future, it uses a relatively high learning rate ($\eta = 0.05$) and a relatively low discount factor ($\gamma = \frac{3}{4}$). Which entries in the Q-table have changed, after this learning, and what are their new values?

Solution:

- $t = 0$:

$$q_{local} = r_0 + \gamma \max_a q(s_1, a) = 2 + 0 = 2$$
$$q(1, \text{walk}) \leftarrow q(1, \text{walk}) + \eta (q_{local} - q(1, \text{walk}))$$
$$= 0 + 0.05(2 - 0) = 0.1$$

- $t = 1$:

$$q_{local} = r_1 + \gamma \max_a q(s_2, a) = 5 + 0 = 5$$
$$q(2, \text{purr}) \leftarrow q(2, \text{purr}) + \eta (q_{local} - q(2, \text{purr}))$$
$$= 0 + 0.05(5 - 0) = 0.25$$

So the changed values are $q(1, \text{walk}) \leftarrow 0.1$ and $q(2, \text{purr}) \leftarrow 0.25$.

(b) (2 points) Instead of model-free learning, the cat decides to implement model-based learning. It estimates $P(s'|s,a)$ using Laplace smoothing, with a smoothing parameter of $k = 1$, using the two SARS observations listed at the start of this problem. What are the new values of $P(s'|s = 2, a = \text{purr})$ for $s' \in \{1, 2\}$?

Solution:

$$P(s' = 1|s = 2, a = \text{purr}) = \frac{1 + \text{Count}(s_t = 2, a_t = \text{purr}, s_{t+1} = 1)}{2 + \sum_{s'} \text{Count}(s_t = 2, a_t = \text{purr}, s_{t+1} = s')} = \frac{1}{3}$$
$$P(s' = 2|s = 2, a = \text{purr}) = \frac{1 + \text{Count}(s_t = 2, a_t = \text{purr}, s_{t+1} = 2)}{2 + \sum_{s'} \text{Count}(s_t = 2, a_t = \text{purr}, s_{t+1} = s')} = \frac{2}{3}$$

(c) (3 points) After many rounds of model-based learning, the cat has deduced that $R(1) = 2$, $R(2) = 5$, and $P(s'|s,a)$ has the following table:

a:	purr		walk	
s:	1	2	1	2
$P(s' = 1\|s,a)$	2/3	1/3	1/3	2/3
$P(s' = 2\|s,a)$	1/3	2/3	2/3	1/3

The cat decides to use policy iteration to find a new optimal policy under this model. It starts with the following policy: $\pi(1) = \text{purr}$, $\pi(2) = \text{walk}$. Now it needs to find the policy-dependent utility, $u^\pi(s)$. Again, because the cat doesn't care about the distant future, it uses a relatively low discount factor ($\gamma = 3/4$). Write two linear equations that can be solved to find the two unknowns $u^\pi(1)$ and $u^\pi(2)$; your equations should have no variables in them other than $u^\pi(1)$ and $u^\pi(2)$.

Solution: The two equations are

$$u^\pi(1) = R(1) + \frac{3}{4}\sum_{s'} P(s'|1,\pi(1))u^\pi(s')$$
$$u^\pi(2) = R(2) + \frac{3}{4}\sum_{s'} P(s'|2,\pi(2))u^\pi(s')$$

Plugging in the given values of all variables, we have

$$u^\pi(1) = 2 + \frac{3}{4}\left(\frac{2}{3}u^\pi(1) + \frac{1}{3}u^\pi(2)\right)$$
$$u^\pi(2) = 5 + \frac{3}{4}\left(\frac{2}{3}u^\pi(1) + \frac{1}{3}u^\pi(2)\right)$$

(d) (2 points) Since it has some extra time, and excellent python programming skills, the cat decides to implement deep reinforcement learning, using a policy gradient algorithm. Inputs are one-hot encodings of state and action. What are the input and output dimensions of the policy network?

Solution: The policy network takes a state as input, thus its input dimension is 2 (if the input is a one-hot encoding of two states). It computes the probability that any given action is the best action, so its output dimension is 2 (if there are two possible actions).

	Simplified GridWorld Column Number	
Row Number	1	2
1	-0.04	0.00
2	-0.04	-1.00
3	1.00	-0.04

Question 9 ***(0 points)***

Consider a simplified version of GridWorld, shown above. Assume that the reward for each state, $r(s)$, is known, and is shown in the map above, but that the transition probabilities $P(s'|s,a)$ are not known. The robot starts in the state with $r(s) = 0.00$; if it reaches either the state with $r(s) = 1.00$ or $r(s) = -1.00$, the game ends.

Let the action variable, a, denote the state to which the robot is trying to move. Assume that, from any state s, for any action a, the possible outcomes s' are only the horizontal and/or vertically neighboring states or the same state ($s' \in \{s, \text{NEIGHBORS}(s)\}$), but the probabilities of these outcomes are unknown.

The robot performs the following action:

- Starting state s: the state with $r(s) = 0.00$).
- Action a: robot tries to move to the horizontally neighboring state.
- Ending state s': the move is successful.

Given this one training observation, use Laplace smoothing, with a smoothing parameter of $k = 1$, to estimate the value of $P(s'|s,a)$ for this particular combination of (s,a,s').

Solution: The starting state is $s = (1,2)$. According to the problem description, the possible outcomes are known to be $s \in \{(1,1),(1,2),(2,2)\}$, so Laplace smoothing gives:

$$P(s' = (1,1)|s = (1,2), a = L) = \frac{1+k}{1+3k}$$
$$= \frac{2}{4}$$

Question 10 ***(0 points)***

A cat lives in a two-room apartment; its current state is given by the room number it currently occupies ($s \in \{1,2\}$). It has two possible actions: walk, or purr. The cat attempts to determine the optimum policy using Q-learning. It starts out with an empty Q-table ($q(s,a) = 0$ for all s and a). Starting in state $s_1 = 1$, it receives the following rewards, performs the following actions, and observes the following resulting states:

t	s_t	r_t	a_t	s_{t+1}
1	1	2	purr	1
2	1	2	purr	1

The cat performs one iteration of time-difference Q-learning with each of these two observations, using a learning rate of $\eta = 0.1$ and a discount factor of $\gamma = 1$.

(a) After these two iterations of Q-learning, what values in the Q-table have changed?

Solution: The cat has only observed the (s,a) combination (1,purr), so the only entry in the Q table that has changed is $q(1,\text{purr})$.

(b) After these two iterations of Q-learning, what is $q(1,\text{purr})$?

Solution: Using the formulas

$$q_{\text{local}}(s,a) = R(s) + \gamma q_t(s',a)$$
$$q_t(s,a) = q_{t-1}(s,a) + \eta\left(q_{\text{local}}(s,a) - q_{t-1}(s,a)\right)$$

We get the following:

$$q_{\text{local}}(1,\text{purr}) = 2+0$$
$$q_1(1,\text{purr}) = 0+0.1\,(2-0) = 0.2$$
$$q_{\text{local}}(1,\text{purr}) = 2+0.2$$
$$q_2(1,\text{purr}) = 0.2+0.1\,(2.2-0.2) = 0.4$$

So the final value is 0.4.

Question 11 ***(0 points)***

Consider an MDP with two actions ($a \in \{R, L\}$), and three states ($s \in \{0, 1, 2\}$). The agent has been accumulating its experiences in an experience replay buffer. The buffer now contains (state,action,reward,new state) (s_t, a_t, r_t, s_{t+1}) tuples for six randomly selected trials, as shown in this table:

Trial ID	s_t	a_t	r_t	s_{t+1}
1	1	R	10	2
2	1	R	10	0
3	0	L	20	1
4	0	L	20	1
5	0	L	20	0
6	2	R	5	1

Using Laplace smoothing with a smoothing coefficient of k, what are $P(s'|s=0, a=L)$ for $s' \in \{0, 1, 2\}$? Write down the general formula for Laplace smoothing, then fill in the numerical values for the problem.

Solution: The general formula for Laplace smoothing is

$$P(X = x|Y = y) = \frac{\text{Count}(X = x, Y = y) + k}{\text{Count}(Y = y) + k|X|},$$

where $|Y|$ is the number of logically distinct possible values of X. For this problem, the results are:

$$P(s' = 0|s = 0, a = L) = \frac{1 + k}{3 + 3k}$$
$$P(s' = 1|s = 0, a = L) = \frac{2 + k}{3 + 3k}$$
$$P(s' = 2|s = 0, a = L) = \frac{k}{3 + 3k}$$

Question 12 ***(0 points)***

Gepetto the toymaker is able to make 0, 1, or 2 wooden toys in any given day. The state of his shop is well summarized by s =the number of toys that he has for sale. On May 8, 2023, he has $s_t = 7$ toys for sale, and earns $r(7) = \$50$. He decides to produce $a_t = 2$ new toys. On May 9, he learns that he now has $s_{t+1} = 8$ toys left in the store; on this day, he decides to produce $a_{t+1} = 0$ new toys. Prior to the May 8 action, he estimated that the quality of each (state,action) pair is as given in the following table:

	$q_t(s,a=0)$	$q_t(s,a=1)$	$q_t(s,a=2)$
$s=7$	25	60	90
$s=8$	10	100	30

Find two different ways in which Gepetto might update $q_t(s,a)$ to compute $q_{t+1}(s=7,a=2)$. Specifically, find $q_{t+1}(s=7,a=2)$ using both TD-learning and SARSA. Both updates should be written as functions of the learning rate, η, and the discount factor, γ; there should be no other variables on the right-hand-side of your answer.

Solution: Using TD-learning and SARSA, respectively, Qlocal is:

$$\textbf{TD Learning:} \quad q_{\text{local}}(s_t,a_t) = r(s_t) + \gamma \max_{a'} q_t(s_{t+1},a')$$

$$\textbf{SARSA:} \quad q_{\text{local}}(s_t,a_t) = r(s_t) + \gamma q_t(s_{t+1},a_{t+1})$$

Plugging in the numbers from this problem, we have

$$\textbf{TD Learning:} \quad q_{\text{local}}(7,2) = 50 + 100\gamma$$

$$\textbf{SARSA:} \quad q_{\text{local}}(7,2) = 50 + 10\gamma$$

The resulting Q-learning updates are

$$\textbf{TD Learning:} \quad q_{t+1}(7,2) = 90 + \eta(50 + 100\gamma - 90)$$

$$\textbf{SARSA:} \quad q_{\text{local}}(7,2) = 90 + \eta(50 + 10\gamma - 90)$$

which can also be written as:

$$\textbf{TD Learning:} \quad q_{t+1}(7,2) = (1-\eta)90 + \eta(50 + 100\gamma)$$

$$\textbf{SARSA:} \quad q_{\text{local}}(7,2) = (1-\eta)90 + \eta(50 + 10\gamma)$$

Question 13 ***(0 points)***

A cat lives in a two-room apartment, and its current state is well-summarized by specifying the room it currently occupies, $s \in \{1,2\}$. In each room, the cat can choose one of two possible actions, $a \in \{\text{pounce}, \text{sleep}\}$. Its current estimates of the quality of each action are:

	$q_t(s, a = \text{pounce})$	$q_t(s, a = \text{sleep})$
$s = 1$	4	6
$s = 2$	8	3

At time t, the cat is in room $s_t = 1$, and receives a reward of $r(1) = 2$ kitty treats. The cat decides to pounce ($a_t = \text{pounce}$), and finds itself in room 2 at time $t+1$. Having found itself in room $s_{t+1} = 2$, the cat decides to sleep ($a_{t+1} = \text{sleep}$). Find two different ways in which the cat might update $q_t(s,a)$ to compute $q_{t+1}(s = 1, a = \text{pounce})$. Specifically, find $q_{t+1}(s = 1, a = \text{pounce})$ using both TD-learning and SARSA. Both updates should be written as functions of the learning rate, η, and the discount factor, γ; there should be no other variables on the right-hand-side of your answer.

Solution: Using TD-learning and SARSA, respectively, Qlocal is:

$$\textbf{TD Learning:} \quad q_{\text{local}}(s_t, a_t) = r(s_t) + \gamma \max_{a'} q_t(s_{t+1}, a')$$

$$\textbf{SARSA:} \quad q_{\text{local}}(s_t, a_t) = r(s_t) + \gamma q_t(s_{t+1}, a_{t+1})$$

Plugging in the numbers from this problem, we have

$$\textbf{TD Learning:} \quad q_{\text{local}}(1, \text{pounce}) = 2 + 8\gamma$$

$$\textbf{SARSA:} \quad q_{\text{local}}(1, \text{pounce}) = 2 + 3\gamma$$

The resulting Q-learning updates are

$$\textbf{TD Learning:} \quad q_{t+1}(7,2) = (1-\eta)4 + \eta(2 + 8\gamma)$$

$$\textbf{SARSA:} \quad q_{\text{local}}(7,2) = (1-\eta)4 + \eta(2 + 3\gamma)$$