Extra Practice for Final

Reinforcement Learning Problem

You are designing an AI system to play a game involving several slot machines. Each turn, the AI must play the slot machine it is at and then it must take an action that moves it to an adjacent location. (Note that, in this problem, the agent is not allowed to repeat a slot machine two times in a row.) The probability distribution for each slot machine is known (see below), and the starting location of the agent is at the left-most slot machine. Each turn, the agent loses $1 as the cost of playing the slot machine. (Note that you are allowed to have a negative score, here.)

Slot 1	Slot 2	Slot 3	Slot 4
$0 – 50%	$0-80%	$0-60%	$0-99%
$1 – 30%	$5 – 20%	$1-30%	$150-1%
$2 – 20%		$4-10%	

Create and fill out a value learning table (with three rounds) for this game, using the same approach as we used in class. Use a value of .9 for γ. (Note that, to find the immediate reward for a state, you can sum up the rewards multiplied by their corresponding probabilities.)

	Slot1	Slot2	Slot3	Slot4
Round 1	0	-0.3	0.5	-0.3
Round 2	-0.27	0.15	0.23	0.15
Round 3	0.135	-0.093	0.635	-0.093

(See calculations below:)

Round 1:
Q(Slot1) – can only move to slot2
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot2 = 1 + 0 – 1 = 0

Q(Slot2) – can move to slot 1 or slot 3
Value of slot 1 = R + max(γ * (future rewards)) - cost
R = .5 * $0 + .3 * $1 + .2 * $2 = $0.70
Value at Slot1 = .7 + 0 – 1 = -.3
OR
Value of Slot3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot3 = .7 + 0 – 1 = -.3

Both options are equally sound, so **either works**.

Q(Slot3) – can move to slot 2 or 4
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot 2 = 1 + 0 – 1 = 0
OR
Value at Slot4 = R + max(γ * (future rewards)) - cost
R = .99 * $0 + .01 * $150 = $1.50
Value at Slot 4 = .7 + 0 – 1 = .50

The best action is to choose **slot4**

Q(Slot4) – can only move to slot 3
Value at Slot 3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot 3 = .7 + 0 – 1 = -.3

Round 2:

Q(Slot1) – can only move to slot2
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot2 = 1 + .9 * -.3 – 1 = -.27

Q(Slot2) – can move to slot 1 or slot 3
Value of slot 1 = R + max(γ * (future rewards)) - cost
R = .5 * $0 + .3 * $1 + .2 * $2 = $0.70
Value at Slot1 = .7 + .9 * -.3 – 1 = -.57
OR
Value of Slot3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot3 = .7 + .9 * .5 – 1 = +.15

The best action is to choose **slot3** (q = +.15).

Q(Slot3) – can move to slot 2 or 4
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot 2 = 1 + .9 * -.3 – 1 = -.27
OR
Value at Slot4 = R + max(γ * (future rewards)) - cost
R = .99 * $0 + .01 * $150 = $1.50
Value at Slot 4 = $1.50 + .9*-.3 – 1 = $0.23

The best action is to choose **slot4** (q = 0.23).

Q(Slot4) – can only move to slot 3
Value at Slot 3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot 3 = .7 + .9 * .5 – 1 = .15

Round 3:

Q(Slot1) – can only move to slot2
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot2 = 1 + .9 * .15 – 1 = .135

Q(Slot2) – can move to slot 1 or slot 3
Value of slot 1 = R + max(γ * (future rewards)) - cost
R = .5 * $0 + .3 * $1 + .2 * $2 = $0.70
Value at Slot1 = .7 + .9 * -.27 – 1 = -.543
OR
Value of Slot3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot3 = .7 + .9 * 0.23 – 1 = -0.093

The best action is to choose **slot3** (q = -0.093).

Q(Slot3) – can move to slot 2 or 4
Value at Slot2 = R + max(γ * (future rewards)) - cost
R = .8 * $0 + .2 * $5 = $1.00
Value at Slot 2 = 1 + .9 * .15 – 1 = .135
OR
Value at Slot4 = R + max(γ * (future rewards)) - cost
R = .99 * $0 + .01 * $150 = $1.50
Value at Slot 4 = $1.50 + .9*.15 – 1 = $0.635

The best action is to choose **slot4** (q = .635).

Q(Slot4) – can only move to slot 3
Value at Slot 3 = R + max(γ * (future rewards)) - cost
R = .6 * $0 + .3 * $1 + .1 * $4 = $0.70
Value at Slot 3 = .7 + .9 * 0.23 – 1 = -0.093

Markov Chain/Hidden Markov Model Problem

Suppose that the menu of the Rat each day differs probabilistically, based on the Markov Model properties. If scones are served on a given day, the probability that they'll be served the next day is 85%. If scones are not served, the probability that they'll be served on the next day is 60%. Use the forward algorithm unless otherwise instructed.

A. Given this information, and given that on Monday scones were served at the Rat, what is the probability that scones will be served on Wednesday? What is the probability that scones are served every weekday (Monday through Friday) this week?

$$\mathbf{P}(\mathbf{scones\ on\ Wednesday}) = \begin{vmatrix}1\\0\end{vmatrix} * \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^2$$

For part 2, you can calculate each probability with the above formula and multiply P(Tues)*P(Wed)*P(Thurs)*P(Fri). However, there's a shortcut here: Since it's asking for the probability of scones being served (given that scones were previously served), we *know* that probability: .85

P(scones on Tuesday – Friday) = $.85^4$ = .522

Now, let's expand the problem a little: If scones are served, the probability of cookies also being served is 5%. If scones are not served, the probability of cookies being served is 80%.

B. If you see people on campus eating cookies on Wednesday and Thursday (and not on Tuesday or Friday), what is the likelihood that scones were served in the cafeteria on those days, using the forward algorithm?

$$\mathbf{P}(\mathbf{scones\ on\ Tuesday}) = \begin{vmatrix}1\\0\end{vmatrix} * \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.05 & 0\\ 0 & .8\end{vmatrix}$$

$$\mathbf{P}(\mathbf{scones\ on\ Wednesday}) = \mathbf{P}(\mathbf{scones\ on\ Tuesday}) * \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.95 & 0\\ 0 & .2\end{vmatrix}$$

$$\mathbf{P}(\mathbf{scones\ on\ Thursday}) = \mathbf{P}(\mathbf{scones\ on\ Wednesday}) * \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.95 & 0\\ 0 & .2\end{vmatrix}$$

$$\mathbf{P}(\mathbf{scones\ on\ Friday}) = \mathbf{P}(\mathbf{scones\ on\ Thursday}) * \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.05 & 0\\ 0 & .8\end{vmatrix}$$

C. Using the previous problem, how does applying the forward-backward algorithm modify the observed probability of scones for Tuesday?

$$\mathbf{B}(\boldsymbol{Friday}) = \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.05 & 0\\ 0 & .8\end{vmatrix} * \begin{vmatrix}1\\1\end{vmatrix}$$

$$\mathbf{B}(\boldsymbol{Thursday}) = \begin{vmatrix}.85 & .15\\ .6 & .4\end{vmatrix}^1 * \begin{vmatrix}.95 & 0\\ 0 & .2\end{vmatrix} * B(Friday)$$

$$\mathbf{B}(\boldsymbol{Wednesday}) = \begin{vmatrix} .85 & .15 \\ .6 & .4 \end{vmatrix}^{1} * \begin{vmatrix} .95 & 0 \\ 0 & .2 \end{vmatrix} * B(Thursday)$$

$$\mathbf{B}(\boldsymbol{Tuesday}) = \begin{vmatrix} .85 & .15 \\ .6 & .4 \end{vmatrix}^{1} * \begin{vmatrix} .05 & 0 \\ 0 & .8 \end{vmatrix} * B(Wednesday)$$

The results of these formulas (Forward and backward) are multiplied together to get the final result; that is left as an exercise for you, if you desire. (Remember that there is still an alpha in these results! You must marginalize this out to get the final probability values.)

Neural Networks problem

In this problem, we are required to create a perceptron that takes in three inputs (with int values ranging from 0 to 10) and returns 1 if the inputs add up to 15 (or more).

Using the initial weights of -1, 2, 3, and 4 and the training samples provided below, what are the weights for the neural network after two epochs? (For this problem, assume we're using a threshold activation function, where the value returned is 1 if the inputs * the weights add up to 0 or more.)

Training Samples

X1	X2	X3	Desired output
8	6	0	0
4	4	9	1
0	8	8	1
5	0	7	0

Solution:

Epoch	Starting weights				Example					Weighted sum	Predict h(x)	Error y – h(x)	Updated weights			
	w0	w1	w2	w3	x0 (bias)	x1	x2	x3	y				w0	w1	w2	w3
1	-1	2	3	4	1	8	6	0	0	33	1	-1	-2	-6	-3	4
1	-2	-6	-3	4	1	4	4	9	1	-2	0	1	-1	-2	1	13
1	-1	-2	1	13	1	0	8	8	1	111	1	0	-1	-2	1	13
1	-1	-2	1	13	1	5	0	7	0	80	1	-1	-2	-7	-7	6
2	-2	-7	-7	6	1	8	6	0	0	-98	0	0	-2	-7	-7	6
2	-2	-7	-7	6	1	4	4	9	1	-4	0	1	-1	-1	-3	15
2	-1	-1	-3	15	1	0	8	8	1	95	1	0	-1	-1	-3	15
2	-1	-1	-3	15	1	5	0	7	0	99	1	-1	-2	-3	-11	7

Weight update examples:

$W[i_{t+1}] = w[i] + ((y-h(x)) * x[i])$

Round 1

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -1 \\ 2 \\ 3 \\ 4 \end{pmatrix} + (0-1)\begin{pmatrix} 1 \\ 8 \\ 6 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 \\ -6 \\ -3 \\ 4 \end{pmatrix}$$

Round 2

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -2 \\ -6 \\ -3 \\ 4 \end{pmatrix} + (1-0)\begin{pmatrix} 1 \\ 4 \\ 4 \\ 9 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ 1 \\ 13 \end{pmatrix}$$

Round 3

-No changes

Round 4

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -2 \\ 1 \\ 13 \end{pmatrix} + (0-1)\begin{pmatrix} 1 \\ 5 \\ 0 \\ 7 \end{pmatrix} = \begin{pmatrix} -2 \\ -7 \\ -7 \\ 6 \end{pmatrix}$$

Round 5

-No changes

Round 6

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -2 \\ -7 \\ -7 \\ 6 \end{pmatrix} + (1-0)\begin{pmatrix} 1 \\ 4 \\ 4 \\ 9 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \\ -3 \\ 15 \end{pmatrix}$$

Round 7

-No changes

Round 8

$$\begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \\ -3 \\ 15 \end{pmatrix} + (0-1)\begin{pmatrix} 1 \\ 0 \\ 8 \\ 8 \end{pmatrix} = \begin{pmatrix} -2 \\ -3 \\ -11 \\ 7 \end{pmatrix}$$

You should be prepared for problems relating to the first half of the semester, including:

- Bayes nets and probabilistic reasoning
- Djikstra's algorithm, A* algorithm, greedy best-first search
- Minimax (with both alpha/beta pruning and with heuristics)
- Which approach that we've talked about so far is best for a specific context and why
- Heuristic design (including consistent/admissible), problem setup, and discussion about the state space.

You should also be prepared for problems we've worked on since the first exam:

- Statistical inference and naïve Bayes classifiers
- Markov Chains and Hidden Markov Models
- Reinforcement Learning strategies, including q-learning and v-learning algorithms
- Neural networks

This exam will be comprehensive-I recommend working through the problems from this guide and from the previous review guide on the course website. The homeworks (and homework solutions) should also be very helpful in preparing for this exam.

For this exam, you are allowed two pages of handwritten notes (front and back). Calculators are not allowed for this exam.