1 Game Tree Search (10 points)

Figure 1 shows the game tree of a two-player game; the first player is the maximizer and the second player is the minimizer. Use the tree to answer the following questions.

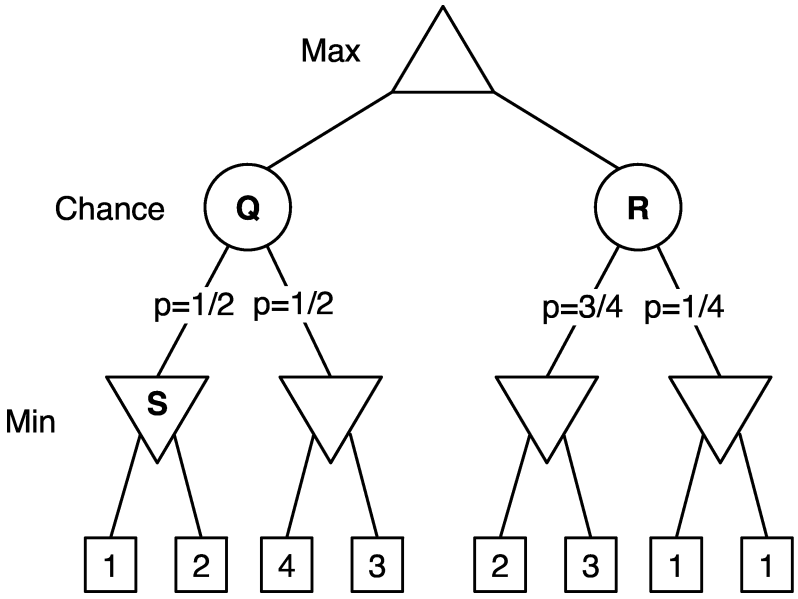


Figure 1: Game tree of two-player game with chance

1. **Circle one (2 pts) :** What is the value of the node labeled S?
   Answer: (b)

   (a) 1/2
   (b) 1
   (c) 2
   (d) Not enough information / cannot be determined

2. **Circle one (2 pts):** What is the expected value of the node labeled Q?
   Answer: (d)

   (a) 1/2
   (b) 1
   (c) 3/2
   (d) 2
   (e) 3
   (f) Not enough information / cannot be determined

3. **Circle one (2 pts):** What is the expected value of the node labeled R?
   Answer: (b)

   (a) 2
   (b) 7/4
   (c) 1
   (d) 2/3
   (e) 1/4
   (f) Not enough information / cannot be determined

4. **Circle one (2 pts):** What is the expected value of the game?
   Answer: (c)

   (a) 1
   (b) 7/4
   (c) 2
   (d) 3
   (e) Not enough information / cannot be determined

5. **True or False (2 pts):** You have been provided with enough information so that you could modify the alpha-beta pruning algorithm to work on this game tree.
   Answer: True

Circle one: Which of the above statements are true?
Answer: (b)

(a) I only
(b) II only
(c) III only
(d) I and II
(e) I and III
(f) Neither I, II, nor III

2 Uninformed Search (10 points)

This question is about a search algorithm called **iterative lengthening search**. We haven't talked about this algorithm in class - it is not the same thing as iterative deepening search. However, we think you can reason about it based on the uninformed search strategies you have already seen.

Iterative lengthening search is an iterative version of uniform cost search. The main idea of this algorithm is that we use increasing limits on path cost.

- We start searching as though we were performing uniform cost search.
- If we generate a node with a path cost that is greater than the current limit, we immediately discard that node.
- We set the limit for the current iteration to be equal to the lowest path cost of any node that we discarded in the previous iteration.
- We initialize the path cost limit to be equal to the smallest cost between any two nodes in the tree. So, for a uniform cost tree, we initialize the path cost limit to 1.

1. **True or False (3 pts):** When this algorithm first encounters the goal, the goal will be on the path with the cheapest cost.
   Answer: True

2. **Circle one (3 pts):** Suppose we are searching through a uniform tree with branching factor b, solution depth d, and unit step costs (that is, the cost to move between two nodes is always one). In the worst case, how many iterations will iterative lengthening require?
   Answer: (b)

   (a) b iterations
   (b) d iterations
   (c) b^d iterations
   (d) d^b iterations
   (e) Not enough information / cannot be determined

3. (More Difficult) (4 pts) Now suppose that we are using the same tree as in the previous question, but our step costs are drawn from the continuous range $[0, 1]$ with a minimum positive cost ϵ. Our path cost limit will be initialized to ϵ. Consider the following statements:

   - I. Iterative lengthening will not find the optimal solution for this tree.
   - II. In the worst case, running iterative lengthening on this tree will require less time than running DFS on this tree.
   - III. In the worst case, iterative deepening search would require less time to run on this tree than iterative lengthening.

3 Decision Trees (14 points)

Your spaceship has just landed on an alien planet, and your crew has begun investigating the local wildlife. Unfortunately, most of your scientific equipment is broken, so all you can tell about a given object is what color it is, how many eyes it has, and whether or not it is alive. To make matters worse, none of you are biologists, so you are going to have to use a decision tree to classify objects near your landing site as either alive or not alive. Use the table below to answer the following questions:

Object	Color	Number of eyes	Alive
A	Red	4	Yes
B	Black	42	No
C	Red	13	Yes
D	Green	3	Yes
E	Black	27	No
F	Red	2	Yes
G	Black	1	Yes
H	Green	11	No

1. **Circle one (2 pts):** Which of the following is the largest? (Note that we are not asking for exact values. You may solve this problem by simply inspecting the table.)
   Answer: (c)

   (a) H(Alive|Number of eyes > 10)
   (b) H(Alive|Number of eyes < 5)
   (c) H(Alive|Color = Green)
   (d) H(Alive|Color = Black)

2. **Fill in the following blank (2 pts):**

   What is the entropy of Alive?
   Answer: H(Alive) $= -\frac{5}{8}log_2\frac{5}{8} - \frac{3}{8}log_2\frac{3}{8} = 0.9544$

3. **Fill in the following blank (2 pts):**

   What is IG(Alive|Color)?
   Answer: 0.36

4. **Circle one (2 pts):** Suppose we wanted to turn Number of eyes into a binary attribute for the purpose of building a decision tree. Which of the following binary categorial splits results in the larger value of IG(Alive—Number of eyes)? (Note that we are not are not asking for exact values. You may solve this problem by simply inspecting the table.)
   Answer: (b)

(a) {Number of eyes = 11, Number of eyes ≠ 11}

(b) {Number of eyes ≤ 4, Number of eyes > 4}

(c) {Number of eyes ≤ 13, Number of eyes > 13}

5. (6 pts) Suppose we were going to build a decision tree for this data:

   - First, we split using the attribute you chose in the previous question.
   - Second, we split on Color.

   How would this tree classify the following objects? (In case of a tie at a leaf node, classify the object as Not alive.) NOTE: This should not be a very complicated tree.

   (a) Circle one (3 pts): **(Alive or Not alive)** A red object with 23 eyes

   (b) Circle one (3 pts): **(Alive or Not alive)** A black object with 1.5 eyes

Answer:

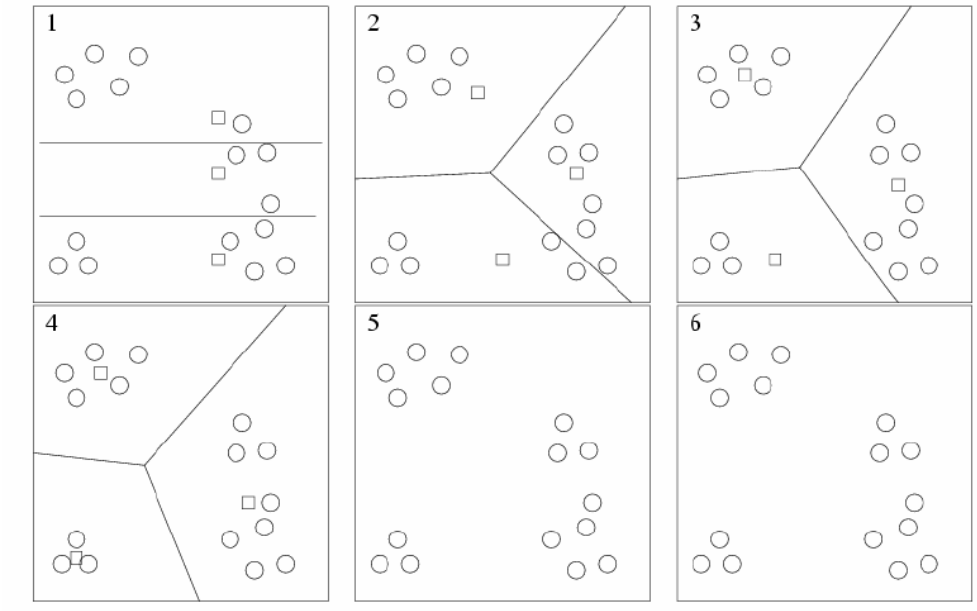
If (a) on previous question, Not Alive; Not Alive

If (b) on previous question, Alive; Alive

If (c) on previous question, Not Alive ; Alive

5 K-Means (7 points)

Run K-means manually on the following dataset. Trace through the first six iterations of the K-means algorithm or until convergence is reached. Circles are data points and squares are the initial cluster centers. Draw the cluster centers and the approximate decision boundaries that define each cluster. (NOTE: It is not necessary to draw the exact location of the squares, but it should be clear from your placement of the squares that you understand how K-means performs quantitatively.)



4 Naive Bayes (10 points)

For each question below, you are **required to write down the basic formulas that you use to compute your answers**. Otherwise, you can only get a maximum of half credit.

Tom is a CMU student. Recently, his mood has been highly influenced by two factors: the weather (W) and his study (S). Naturally, he likes good weather and hates bad weather. More importantly, Tom worries about his exams. Tom feels happy if he passes exams and not happy if he fails them. Now Tom wants to predict his happiness according to these two factors using his previous experience. Tables A and B show this data.

Weather(W)	Study(S)	Happy(H)
Bad	Fail	0
Good	Fail	0
Good	Fail	0
Good	Fail	0
Bad	Pass	0
Bad	Pass	1
Bad	Pass	1
Good	Pass	1

Table A: 2 factors

(a) Using Table A: If today's situation is W=Good, S=Pass, and Tom uses a Naive Bayes classifier, how would he predict his happiness? Please show your computations and the classifier's prediction. (2 pts)

Answer:
$P(W = G|H = 0)P(S = P|H = 0)P(H = 0) = 3/40$
$P(W = G|H = 1)P(S = P|H = 1)P(H = 1) = 1/8$
predict Happy.

(b) Using Table A: If today's situation is W=Bad, S=Fail, and Tom uses a Naive Bayes classifier, how would he predict his happiness? Please show your computations and the classifier's prediction. (2 pts)

Answer:
$P(W = B|H = 0)P(S = F|H = 0)P(H = 0) = 1/5$
$P(W = B|H = 1)P(S = F|H = 1)P(H = 1) = 0$
predict Unhappy.

Tom also notices that his neighbor always goes for a walk if the weather is good and stays at home if the weather is bad. Tom thinks it wouldn't hurt to have more information, so he adds one more

factor, Neighbor (N), to the table. The new table is shown as Table B. You can see that whenever W=Good, N=Out, and whenever W=Bad, N=home.

Weather(W)	Study(S)	Neighbor(N)	Happy(H)
Bad	Fail	Home	0
Good	Fail	Out	0
Good	Fail	Out	0
Good	Fail	Out	0
Bad	Pass	Home	0
Bad	Pass	Home	1
Bad	Pass	Home	1
Good	Pass	Out	1

Table B: 3 factors

(c) Using Table B: Now, if W=Good, S=Pass, N=Out, and Tom uses a Naive Bayes Classifier, how would he predict his happiness? Please show your computations and the classifier's prediction. (2 pts)

Answer:
$P(W = G|H = 0)P(S = P|H = 0)P(H = 0) = 9/200$
$P(W = G|H = 1)P(S = P|H = 1)P(H = 1) = 1/24$
predict Unhappy.

(d) Will the new factor improve the performance of the Naive Bayes classifier? Why or why not? (2 pts)

Answer: No. Since weather and neighbor are not conditionally independent. The assumption of Naive Bayes doesn't hold anymore.

(e) Using Table B: Now, if Tom uses a Bayes Classifier instead of a Naive Bayes Classifier, and we still assume W=Good, S=Pass, N=Out, how would he predict his happiness? Please show your computations and the classifier's prediction. (2 pts)

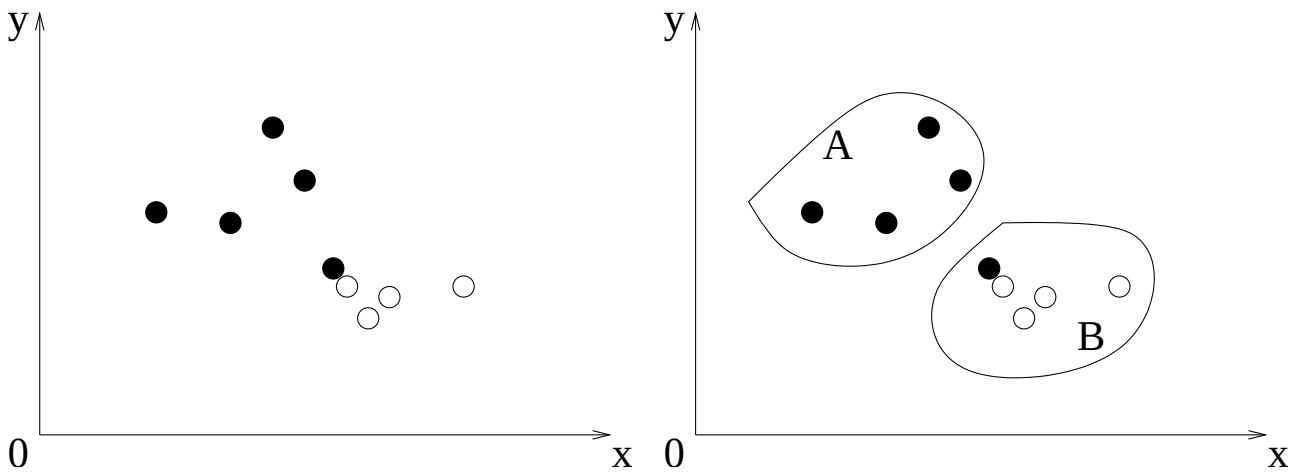
Answer:
$P(W = G, S = P, N = O|H = 0)P(H = 0) = 0$
$P(W = G, S = P, N = O|H = 1)P(H = 1) = 1/8$
predict Happy.

6 Cross Validation (10 points)

K-nearest-neighbor is a very simple classification algorithm. It works like this:

- We wish to classify a new data point q.
- We compute the distance from q to every data point in the training dataset. We typically use Euclidian distance.
- We find the K closest data points to q.
- We classify q by giving it the same class as the majority of those K closest data points (q is classified as the majority class).

Now, suppose you are running a K-nearest-neighbor classifier on the following training set. The training set is shown below. It consists of 9 data points. The black dots have class label 1, and the white dots have class label 0.



(a) Use the figure on the left: If we use 1-nearest-neighbor, what is the leave-one-out Cross-Validation error? (Report the error as a ratio.) (2 pts)
Answer: $\frac{2}{9}$

(b) Use the figure on the left: If we use 3-nearest-neighbor, what is the leave-one-out Cross-Validation error? (Report the error as a ratio.) (2 pts)
Answer: $\frac{1}{9}$

(c) Use the figure on the right: What is the two-fold Cross-Validation error for 1-nearest-neighbor? Assume we separate the data into two sets, A and B, as shown in the figure. (Report the error as a ratio.) (3 pts)
Answer: $\frac{4}{9}$

(d) Use the figure on the right: What is the two-fold Cross-Validation error for 3-nearest-neighbor? Assume we separate the data into two sets, A and B, as shown in the figure. (Report the error as a ratio.) (3 pts)
Answer: $\frac{8}{9}$

7 Bayes Nets (10 points)

Given the Bayes net shown in the Figure below; A, B, C, D, and E are all Boolean variables. P(A="+") is simply denoted as P(A).

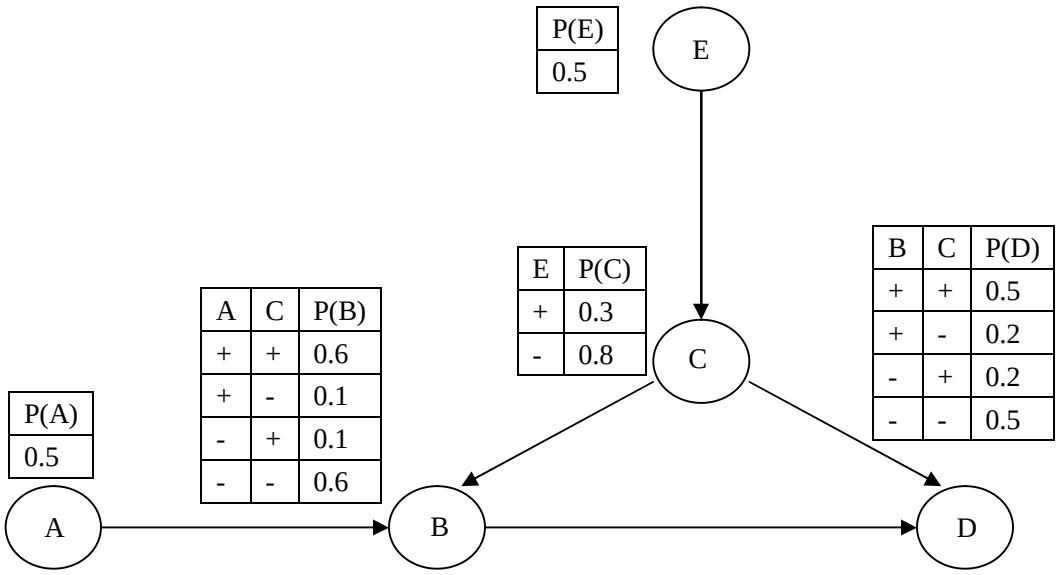


Figure 2: Bayes Net

Note: In the following, we'll use the notations: $A \perp B$ means A is independent of B; $A \perp B|C$ means A is conditionally independent of B given C.

(a) Please judge if the following independence assumptions are correct or not:

1. **(True or False) (1 pt):** $B \perp E|C$
   Answer: True
2. **(True or False) (1 pt):** $A \perp D$
   Answer: False
3. **(True or False) (2 pts):** $A \perp D|B$
   Answer: False
4. **(True or False) (2 pts):** $A \perp D|B, C$
   Answer: True

(b) Compute the value of $P(C)$ (2 pts)
Answer: 0.55

(c) Compute the value of $P(B|A)$ (2 pts)
Answer: 0.375

8 Markov Decision Process (9 points)

In the following Markov Decision Process, there are three states S_1, S_2 and S_3. The rewards for each state and all of the state transitions are marked in the given figure. There are two actions. Action a_1 causes you to stay in the same state, and action a_2 causes you to move to other states. The discount factor is $\gamma = 0.9$.

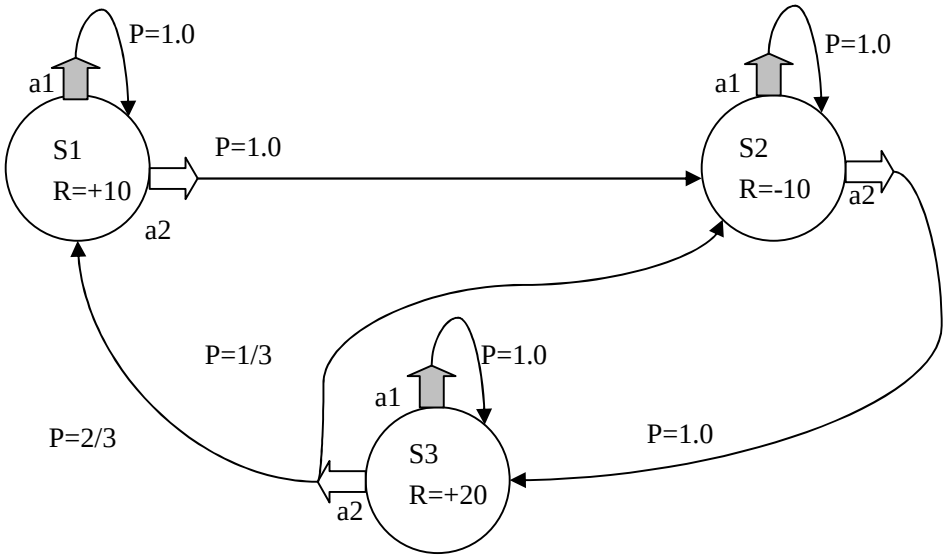


Figure 3: Markov Decision Process

(a) How many distinct policies are there in the above MDP? (1 pt)
Answer: $2^3 = 8$

(b) $U^{\pi_0}(i)$ is the expected sum of discounted rewards if we start at state i and follow the policy π_0. The initial policy π_0 is the one that assigns a_1 to every state. Write down the numerical values of the expected discounted rewards of each of the following states in the MDP. (3 pts)

1. $U^{\pi_0}(S_1) = 100$
2. $U^{\pi_0}(S_2) = $ -100
3. $U^{\pi_0}(S_3) = 200$

(c) Continuing from part (b), suppose we run policy iteration with π_0 as the initial policy. Define π_1 as the updated policy after one iteration of policy iteration, and write down the updated policy for each of the states (2 pts):

1. $\pi_1(S_1) = a_1$
2. $\pi_1(S_2) = a_2$
3. $\pi_1(S_3) = a_1$

(d) Suppose we run value iteration, $U^k(i)$ is the expected sum of discounted rewards if we start at state i after $(k-1)$-step of value iterations. Starting from the initial value of $U^1(i)$, which is the reward at state i, please write down the updated $U^2(i)$ after one value iteration (3 pts).

1. $U^2(S_1) = 19$
2. $U^2(S_2) = 8$
3. $U^2(S_3) = 38$

9 Neural Network (12 points)

Consider a problem in which each data point has two coordinates u and v. We wish to learn a classifier for this problem by using a linear perceptron network with inputs u and v and weights W_u and W_v on the two connections. We use a threshold of 0 so that the output of the network is +1 (class 1) if the output unit is greater than or equal to 0 and -1 (class 2) otherwise.

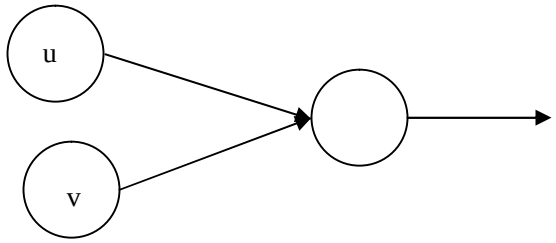
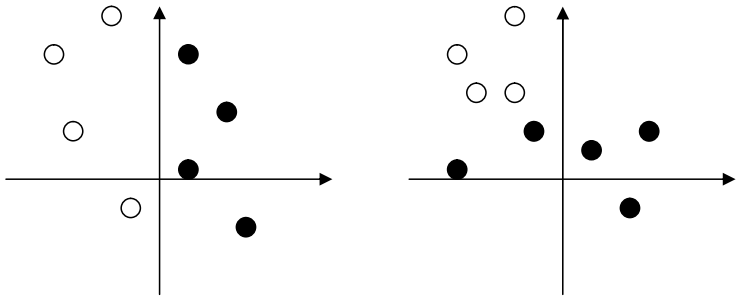


Figure 4: Neural Net

(a) Can the network distinguish the two classes in the cases illustrated below? Why/Why not in each case? (2 pts)



Case I Case II

Answer:
Yes for case 1 and no for case 2. Without a constant term, the decision boundary must go through the origin.

(b) Assume that we augment the network with an additional input unit with constant value 1 (we'll call the weight on the additional connection W). How does the answer to 1 change and why? (2 pts)

Answer:
With a constant term the decision boundary can go anywhere though it must still be linear. So both cases I and II can now be discriminated correctly.

(c)(More difficult)The sigmoid perceptron defines Prediction = $g(w_0 + w_1x_1 + w_2x_2 + \ldots w_kx_k)$, given k input variables x_1 through x_k, and $g(z) = \frac{1}{1+exp(-z)}$. Pat thinks the sigmoid function is too computationally expensive and replaces it with this piecewise linear approximation: Prediction =

$a(w_0 + w_1x_1 + w_2x_2 + \ldots w_kx_k)$, where

$$a(z) = \begin{cases} 0, & \text{if} \quad z < -1 \\ (z+1)/2, & \text{if} \quad -1 \leq z \leq 1 \\ 1, & \text{if} \quad z > 1 \end{cases} .$$

Pat is very pleased with the computational savings, but there is a serious problem with this idea. Please explain the problem. If you wish, you may use the diagram of $a(z)$ below to help you explain the problem with Pat's idea. . (3 pts)

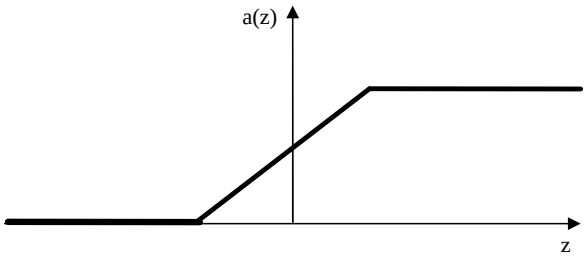


Figure 5: The piecewise linear function of $a(z)$

Answer:
The problem is that a(z) is undifferentiable at two places, which will hurt gradient descent. Worse, it has zero derivative over a large range, and so gradient descent will have no information about which direction to alter the weights.

Consider the following function of two variables:

X	Y	Output
0	0	1
1	0	0
0	1	0
1	1	1

(d) Please draw a simple diagram and use one or two sentences to illustrate that this function cannot be learned by a single linear perceptron. (3 pts)

Answer:

1 0
0 1

There is no single straight line that discriminate the 1's from the 0's

(e) (**True/False**) Would adding one hidden unit to the model allow to learn the above described function? (2 pts)

Answer: False
At least two hidden units are needed. One hidden unit would merely provide an extra nonlinearity on the output.

10 Reinforcement learning (8 points)

Part I.

Consider an agent starting in a room A in which it can take two possible actions: to leave the room (action "L") or to stay (action "S"). If it leaves A, the agent moves to room B, which is a terminal state (no more actions can be taken). The outcomes of the actions are uncertain, so that when executing action L (or action S), there is some probability that the agent will leave A (or stay in A). We assume that the reward in entering state B is $R(B) = +1$ and the reward for being in state A is $R(A) = -0.1$.

(a) Draw the (very simple) diagram corresponding to this MDP. Answer by inspection of the diagram: What is the optimal policy? (2 pts)

Answer: With the added conditions on the probabilities, the optimal policy is $\pi(A) = L$.

(b) Assume that the agent knows neither the world (transition probabilities) nor the utilities of the states. Assume that the agent, for some reason, happens to follow the optimal policy. The rewards received at states A and B are the same as described above.. In the process of executing this policy, the agent execute four trials and, in each trial, it stops after reaching state B. The following state sequences are recorded during the trials: $AAAB$, AAB, AB, AB. What is the estimate of $T(.,.,.)$? What is the estimate of $U(A)$, assuming a discount factor of $\gamma = 0.5$? (2 pts)

Answer:
$T(A, L, A) = 3/7$ and $T(A, L, B) = 4/7$
Note that $T(A, S, A)$ cannot be computed from the data given in the text and it is not needed since we assume that we follow the optimal policy.
$U(A) = R(A) + \gamma(T(A, L, A)U(A) + T(A, L, B)U(B))$
$U(A) = -0.1 + 0.5 \times (3/7 \times U(A) + 4/7 \times 1)$
$11/14 \times U(A) = -0.1 + 4/14$
$U(A) = 26/110 = 0.2364$

(c) Assume now that the agent is executing only one trial yielding the sequence of states AAB. Compute the estimate of the utility $U(A)$ using TD (temporal differencing). Use discount $\gamma = 0.5$, and learning rate $\alpha = 0.5$. (2 pts)

Answer:

Transition A to A:
$U^{new}(A) = U^{old}(A) + \alpha(R(A) + \gamma U^{old}(A) - U^{old}(A))$
$U^{new}(A) = -0.1 + 0.5 \times (-0.1 + 0.5 \times -0.1 - (-0.1)) = -0.125$
Transition A to B:
$U^{new}(A) = U^{old}(A) + \alpha(R(A) + \gamma U(B) - U^{old}(A))$
$U^{new}(A) = -0.125 + 0.5 \times (-0.1 + 0.5 \times 1 - (-0.125)) = 0.1375$
Note that the question did not specify the starting values for U. Alternative solutions (e.g., with U = 0) were also accepted as long as the formulas were correct.

Part II.

We are using Q-learning to learn a policy in an MDP with two states S_1 and S_2 and two actions a and b. Assume that $\gamma = 0.8$ and $\alpha = 0.2$, and that the current values of Q are:

Q(S_1,a)	2.0
Q(S_1,b)	2.0
Q(S_2,a)	4.0
Q(S_2,b)	2.0

Suppose that, when we were in state S_1, we took action b, received reward 1.0 and moved to state S_2. Which item of the Q table will change and what is the new value? (2 pts)

Answer:
$Q(S_1, b)$ is the affected entry.
$Q^{new}(S_1, b) = Q^{old}(S_1, b) + \alpha(R(S_1) + \gamma Max_{action} Q(S_2, action) - Q^{old}(S_1, b))$
$Q^{new}(S_1, b) = 2 + 0.2 \times (1 + 0.8 \times 4 - 2) = 2.44$
Note: A common mistake is to forget the "Max" and to use 0.8×2 instead of the correct expression.