Homework 2 (revision 2)

CSCI 360 Spring 2022

This assignment covers **RL, Logistic Regression and Machine Learning, Perceptrons, Neural Networks, Decision Trees, and Bayes Nets and Hidden Markov Models**. It is due on Friday, April 29, 11:59 PM PST.

Submission Instructions: Please submit your solutions on Gradescope. We recommend you type up your solutions using LaTeX but handwritten solutions are also fine. **Make sure that you correctly assign pages to questions when submitting to Gradescope, otherwise you will receive a 0 for unassigned questions.**

DO NOT SUBMIT TO BLACKBOARD.

SUBMIT TO *GRADESCOPE*.

1 Reinforcement Learning [13 pts]

Your robot vacuum cleaner has discovered that it gains rewards as it moves around and cleans up. In each state, it has four possible actions, Go Left, Go Right, Slide or Climb. The Attic and the Basement are terminal states, with no action possible from there.

Below you will find the robot's sequence of actions in the first episode :

t	s_t	a_t	s_{t+1}	r_t
1	Room	Go Right	Kitchen	5
2	Kitchen	Go Left	Room	2
3	Room	Go Right	Room	2
4	Room	Go Right	Kitchen	5
5	Kitchen	Climb	Attic	10

1. (2 points) In model-based reinforcement learning, the transition (*T(s, a, s')*) and reward (*R(s, a, s')*) functions are estimated first. Use the first episode (given above) to fill the following values:

   (a) *R(Kitchen, Go Left, Room)* =

   Solution: 2

   (b) *R(Room, Go Right, Room)* =

   Solution: 2

   (c) *T(Room, Go Right, Kitchen)* =

   Solution: 2/3 (#(Room, Go Right, Kitchen) / #(Room, Go Right))

   (d) *T(Room, Go Right, Room)* =

   Solution: 1/3

   (e) *T(Kitchen, Climb, Attic)* =

   Solution: 1

   (f) *T(Room, Slide, Basement)* =

   Solution: 0

2. (1 point) Now, consider the second episode :

t	s_t	a_t	s_{t+1}	r_t
1	Room	Go Right	Room	2
2	Room	Go Right	Kitchen	5
3	Kitchen	Go Left	Room	2
4	Room	Slide	Basement	1

   After including these samples, which of the calculated values in part (a) will change? (Mention the new updated value as well)

Solution: *T(Room, Go Right, Kitchen)* = 3/5
T(Room, Go Right, Room) = 2/5
T(Room, Slide, Basement) = 1

3. (3 points) Now, onward to model-free learning. What are the following Q-values learned after ONLY the first episode?
Assume that all Q-values initialized as 0. Update the Q value with the latest Q values at each t, using the equation $Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * [R(s,a,s') + \gamma(max_{a'}(Q(s',a'))]$, where $\gamma = 0.5$, $\alpha = 0.5$. Hint: at each t, you should only be updating a single Q value (the one to which the sample applies).

   (a) *Q(Room, Go Right)* =

   Solution: 139/32

   (b) *Q(Kitchen, Go Left)* =

   Solution: 13/8

   (c) *Q(Kitchen, Climb)* =

   Solution: 5

   Here's a table to help you keep track of values:

t	sample	Q(Room,Go Right)	Q(Room,Slide)	Q(Kitchen,Go Left)	Q(Kitchen,Climb)
1	[Room, Go Right, Kitchen, 5]				
2	[Kitchen, Go Left, Room, 2]				
3	[Room, Go Right, Room, 2]				
4	[Room, Go Right, Kitchen, 5]				
5	[Kitchen, Climb, Attic, 10]				

Solution: Perform Q-value update for every sample in the first episode.
$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * [R(s,a,s') + \gamma(max_{a'}(Q(s',a'))]$

t	sample	Q(Room,Go Right)	Q(Room,Slide)	Q(Kitchen,Go Left)	Q(Kitchen,Climb)
1	[Room, Go Right, Kitchen, 5]	5/2	0	0	0
2	[Kitchen, Go Left, Room, 2]	5/2	0	13/8	0
3	[Room, Go Right, Room, 2]	23/8	0	13/8	0
4	[Room, Go Right, Kitchen, 5]	139/32	0	13/8	0
5	[Kitchen, Climb, Attic, 10]	139/32	0	13/8	5

1. Q(Room,GoRight) = 0 + (1/2) * [5 + (1/2)*0] = 5/2
2. Q(Kitchen,GoLeft) = 0 + (1/2) * [2 + (1/2)*max(5/2,0)] = 13/8

3. Q(Room,GoRight) = (1/2)5/2 + (1/2) * [2 + (1/2)*max(5/2,0)] = 23/8
4. Q(Room,GoRight) = (1/2)23/8 + (1/2) * [5 + (1/2)*max(13/8,0)] = 139/32
5. Q(Kitchen,Climb) = 0 + (1/2) * [10 + (1/2)*0] = 5

4. (1 point) Suppose after numerous episodes, we end up with the following Q-values:
Q(Room, Go Right) = 5.1
Q(Room, Slide) = 4.7
Q(Kitchen, Go Left) = 3.4
Q(Kitchen, Climb) = 7.5

   Give the optimal policies :
   $\pi(Room)$ =

   Solution: Go Right

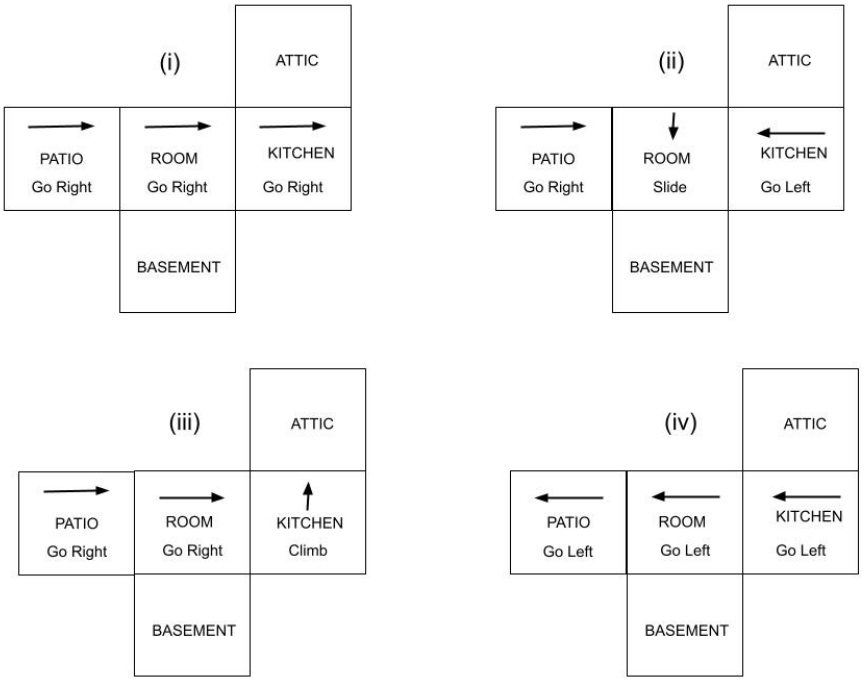
   $\pi(Kitchen)$ =

   Solution: Climb

5. (1 point) Will adding the sample [Room, Slide, Basement, 1] change the optimal policy in 1.4 (assume all Q values in the Basement are 0)? If yes, what will the updated policy be? If no, justify.

   Solution: No change to the optimal policy.
   Q(Room, Slide) = (1/2)*(4.7) + (1/2)*[1 + (1/2)*max(0)] = 2.85
   New Q value *Q(Room, Slide)* = 2.85 which is still lesser than Q(Room, Go Right). The optimal policy will still remain to Go Right when in the Room state.

6. (5 points) Your robot enters a championship with three other participants. Each of them have learnt an optimal policy as shown below, with their own, unspecified reward functions. (i), (ii), (iii) and (iv) are the policies of the four different robots.



Only the transitions to the Attic and Basement have rewards (which can be negative or positive). Transitions to the Kitchen, Room and Patio have no rewards. When a policy takes an action that goes into a wall, the agent will just stay in the same state.

Suppose we use a feature-based representation of the states. The value of each state is $V(x, y) = w^T f(x, y)$. $f(x, y)$ is a feature function which maps the grid coordinates (x, y) to a vector of features and w is a weight vector that parameterizes the value function.

The following few questions consider various possible feature functions $f(x, y)$. **Assume the weight vector w is not allowed to be all 0's. E.g. in 1 dimension, it cannot be 0. In 2 dimensions, at least one of the entries of w must be non-zero.**

(a) Let (x_0,y_0) be the attic's location. If the **x-distance to the attic** is the only feature used, i.e, $f(x,y) = | x - x_0 |$, which of the above policies could be extracted from a value function that is representable using this feature function? Choose all that apply:
(i) (ii) (iii) (iv)

Solution: (i), (iii), (iv)
(ii) is trying to both increase and decrease the x-distance to the attic at the same time.

(b) Let (x_1,y_1) be the basement's location. If we use two features : the x-distance to the basement and the y-distance to the basement. i.e, $f(x,y) = (| x - x_1 | , | y - y_1 |)$, which of the above policies could be extracted from a value function that is representable using this feature function? Choose all that apply:
(i) (ii) (iii) (iv)

Solution: (ii), (iii)
(ii) is decreasing the x-distance and increasing the y-distance to the basement.
(iii) decreases the x-distance and increases the y-distance
(i) and (iv) try to increase and decrease the x-distance to the attic at the same time.

(c) If the **Euclidean distance to the attic** is the only feature used, which of the above policies could be extracted from a value function that is representable using this feature function? Choose all that apply:
(i) (ii) (iii) (iv)

Solution: (iii), (iv)
(i) decreases distance in PATIO and ROOM but doesn't decrease distance at KITCHEN. (ii) increases and decreases the shortest distance to the attic.

2 Logistic Regression and Machine Learning [13 pts]

1. (6 points) **Maximum Likelihood Estimation**: As seen in lectures 8 and 9 and in the discussion for lecture 9, we maximize the likelihood (or log-likelihood) of some function in machine learning to obtain the best estimate of the parameters of our hypothesis (where the hypothesis class is the class of neural networks, logistic regression classifiers, etc.) to classify the given training datapoints. Here, we will practice performing Maximum Likelihood Estimation (MLE) again, on a simpler distribution. If you haven't yet, we highly recommend you watch the discussion for lecture 9 (Lecture on 3/30/22) and read the handout before attempting this problem.

   Assume you have N **independent, identically drawn (iid)** outcomes, $y^{(1)}, ..., y^{(N)}$ from flipping a (possibly biased) coin N times. A coin flip results in either Heads or Tails. We would like to estimate the parameter θ, which represents the true chance that the coin lands on Heads. The likelihood of our sample given our estimate θ is then:

$$P(y^{(1)}...y^{(N)}; \theta) = \Pi_{i=1}^{N} P(y^{(i)}; \theta) \quad \text{(from iid assumption)}$$

   Thus, we would like to maximize this probability over all possible θ to obtain the best estimate.

$$\max_{\theta} \Pi_{i=1}^{N} P(y^{(i)}; \theta) =$$
$$\max_{\theta} \mathcal{L}(\theta)$$

   where $\mathcal{L}(\theta) = \Pi_{i=1}^{N} P(y^{(i)}; \theta)$ is called the "likelihood function." The process of maximizing this function is called **Maximum Likelihood Estimation (MLE)**.

   (a) Explicitly write out the probability that coin $y^{(i)}$ turns out to be Heads given our parameter θ, $P(y^{(i)} = H; \theta)$:

   > **Solution:** $P(y^{(i)} = H; \theta) = \theta$

   (b) Explicitly write out the probability that coin $y^{(i)}$ turns out to be Tails given our parameter θ, $P(y^{(i)} = T; \theta)$:

   > **Solution:** $P(y^{(i)} = T; \theta) = 1 - \theta$

   (c) Now, assume that we have observed $M \leq N$ Heads (and therefore $N - M \geq 0$ Tails). Explicitly write out the likelihood $\mathcal{L}(\theta)$ of these observations as a function expressed in terms of θ, M, and N.

   > **Solution:** $\mathcal{L}(\theta) = \theta^M (1 - \theta)^{N-M}$

   (d) Note that this function is simple enough to solve for directly! Remember from calculus that you can find the maximum of a concave function by directly taking the derivative and solving for 0. What is the derivative of $\mathcal{L}(\theta)$ with respect to the parameters θ ($\frac{\partial}{\partial \theta}\mathcal{L}(\theta)$)? **Assume $M = N - 1$, i.e. that you have only 1 observed tail.**
   $\frac{\partial}{\partial \theta}\mathcal{L}(\theta) =$

   > **Solution:** $\frac{\partial}{\partial \theta}\theta^{N-1}(1 - \theta) = \frac{\partial}{\partial \theta}\theta^{N-1} - \frac{\partial}{\partial \theta}N\theta^N = (N-1)\theta^{N-2} - N\theta^{N-1} = \theta^{N-2}(N - 1 - N\theta)$

   (e) Now, set $\frac{\partial}{\partial \theta}\mathcal{L}(\theta) = 0$ and solve for θ! What's the MLE estimate of θ? Intuitively, what does this estimate for θ mean?

Solution:

$$\theta^{N-2}(N-1-N\theta) = 0$$
$$N-1-N\theta = 0$$
$$\theta = \frac{N-1}{N}$$

This is essentially counting the number of heads and dividing by the total number of flips.

(f) With many machine learning hypotheses classes, the objective does not have a closed form solution. For example, in classification we have samples $(\mathbf{x}^{(1)}, y^{(1)}), ..., (\mathbf{x}_N, y^{(N)})$ and we are trying to maximize the probability of the input $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}$ having the observed labels $y^{(1)}, ..., y^{(N)}$ given our parameters $\mathbf{w}$. In logistic regression, this is $P(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})$ where ϕ represents the sigmoid activation function, and the likelihood function $\mathcal{L}(\theta)$ cannot be directly maximized by taking the derivative and setting it equal to 0 as attempting to do so results in an unsolvable equation.

That's why we have to perform **gradient ascent** to maximize the objective instead. Let's pretend that the same is true for this Heads/Tails coin problem. What is the update rule for our parameter θ (instead of the $\mathbf{w}$ above) if we are performing gradient ascent (building on your answer from (d))? Express your answer in terms of θ, an arbitrary learning rate α, and N.

Solution: $\theta \leftarrow \theta + \alpha \frac{\partial}{\partial \theta}\mathcal{L}(\theta) = \theta + \alpha\theta^{N-2}(N-1-N\theta)$

2. (7 points) **Gradient Ascent for Logistic Regression**: In this question, you will perform gradient ascent updates for a few iterations with a logistic regression classifier on the maximum likelihood objective. If you haven't yet, we recommend you watch the lecture for discussion 9 and the gradient ascent video tutorial posted on blackboard before attempting this problem.

Remember that the objective of classification is to maximize the likelihood of the training data, i.e. maximize the probability of predicting the correct label for all of the datapoints in the training set. In logistic regression for binary classification, we have a linear classifier parameterized by a vector w, such that a prediction is obtained by taking its dot product with the input x and sending that value through a sigmoid function to bound the output between $(0, 1)$. This looks like $\phi(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1+e^{-\sum_{i=1}^{d} w_i x_i}}$ when we have d features. We arbitrarily set $P(y = 1|\mathbf{x}; \mathbf{w}) = \phi(\mathbf{w} \cdot \mathbf{x})$, and $P(y = 0|\mathbf{x}; \mathbf{w}) = 1 - \phi(\mathbf{w} \cdot \mathbf{x})$ for the two labels 0 and 1.

Assuming N training points, the objective for logistic regression is derived as follows:

$$\max_{\mathbf{w}} \Pi_{i=1}^{N} P(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = \max_{\mathbf{w}} \Pi_{i=1}^{N} \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})^{y^{(i)}} \left[1 - \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})\right]^{1-y^{(i)}}$$

We then take the log of this objective so that we can get a sum for easier gradient computation. Note that if we take the log of this function, the maximizing w still doesn't change.

$$\max_{\mathbf{w}} \log\left(\Pi_{i=1}^{N} \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})^{y^{(i)}} \left[1 - \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})\right]^{1-y^{(i)}}\right)$$
$$= \max_{\mathbf{w}} \sum_{i=1}^{N} y^{(i)} \log \phi(\mathbf{w} \cdot \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})) \quad (1)$$

We maximize this function through gradient ascent with respect to the weights w. The gradient is given by

$$
\begin{aligned}
&\nabla_{\mathbf{w}} \sum_{i=1}^{N} y^{(i)} \log \phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) + (1-y^{(i)}) \log(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})) \\
&= \sum_{i=1}^{N} \nabla_{\mathbf{w}} y^{(i)} \log \phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) + \nabla_{\mathbf{w}}(1-y^{(i)}) \log(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})) \quad \text{derivative of sums} \\
&= \left[\frac{y^{(i)}}{\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})} - \frac{1-y^{(i)}}{1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})}\right] \nabla_{\mathbf{w}} \phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) \quad \text{using derivative of log} \\
&= \left[\frac{y^{(i)}}{\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})} - \frac{1-y^{(i)}}{1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})}\right] \phi(\mathbf{w}\cdot\mathbf{x}^{(i)})(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) \nabla_{\mathbf{w}} \mathbf{w}\cdot\mathbf{x}^{(i)} \text{ using } \nabla\phi(z) = \phi(z)(1-\phi(z)) \\
&= \left[\frac{y^{(i)}}{\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})} - \frac{1-y^{(i)}}{1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})}\right] \phi(\mathbf{w}\cdot\mathbf{x}^{(i)})(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) \mathbf{x}^{(i)} \text{ using } \nabla_{\mathbf{w}} \mathbf{w}\cdot\mathbf{x} = \mathbf{x} \text{ just like in 1d calculus} \\
&= \left[\frac{y^{(i)} - \phi(\mathbf{w}\cdot\mathbf{x})}{\phi(\mathbf{w}\cdot\mathbf{x}^{(i)})(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)}))}\right] \phi(\mathbf{w}\cdot\mathbf{x}^{(i)})(1-\phi(\mathbf{w}\cdot\mathbf{x}^{(i)}) \mathbf{x}^{(i)} \text{ algebraic manipulation} \\
&= \sum_{i=1}^{N} \left[y^{(i)} - \phi(\mathbf{w}\cdot\mathbf{x}^{(i)})\right] \mathbf{x}^{(i)}
\end{aligned}
$$

Note that the gradient is a vector of dimension d as we are taking a weighted sum of the datapoint vectors $\mathbf{x}$.

For the following parts, assume that we have 2 datapoints in our training set ($\mathbf{x}^{(1)} = [1\ 1]^T, y^{(1)} = 1$) and ($\mathbf{x}^{(2)} = [0\ 0.5]^T, y^{(2)} = 0$). Assume the initial weights are $\mathbf{w} = [0.5\ 1.0]^T$, and that our learning rate $\alpha = 5$. Give exact decimal answers rounded to two decimal points. Feel free to do this problem either by hand (with a calculator) or by writing your own code to calculate this.

(a) What are our initial predictions for the probability of each datapoint $(\mathbf{x}, y)$)? What is the log likelihood of our training set (Eq.1)? Use the natural log.

$P(y^{(1)}|\mathbf{x}^{(1)}; \mathbf{w}) =$

Solution: $\phi(0.5 + 1.0) = 0.82$

$P(y^{(2)}|\mathbf{x}^{(2)}; \mathbf{w}) =$

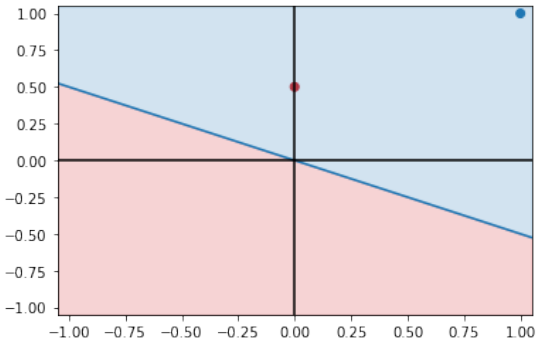
Solution: $1 - \phi(0 + 0.5) = 0.38$

Log Likelihood =

Solution: $\log \phi(0.5 + 1.0) + \log \phi(1 - \phi(0 + 0.5)) = -1.18$

(b) Now, plot the two datapoints and draw a line for the "decision boundary" on a graph. The decision boundary is obtained where the probabilities of predicting either class is equal, or where $P(y|\mathbf{x}; \mathbf{w}) = 0.5$. Hint: for the decision boundary, set $P(y|\mathbf{x}; \mathbf{w}) = 0.5$ and solve for x_2 (the second dimension of $\mathbf{x}$) as a function of w_1, w_2, x_1. Then, plot that line on your graph. Make the graph bounds $[-1, 1]$ for both dimensions.

Solution:



(c) Perform one update of gradient ascent, what is the new **w**? What are the new predicted probabilities of our datapoints and the new log likelihood (natural log)?

w =

Solution: [1.41, 0.36]

$P(y^{(1)}|\mathbf{x}^{(1)}; \mathbf{w})$ =

Solution: 0.85

$P(y^{(2)}|\mathbf{x}^{(2)}; \mathbf{w})$ =

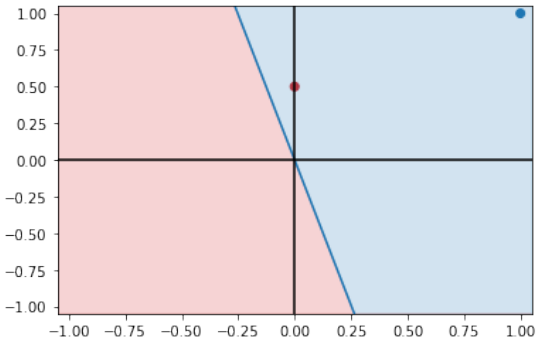
Solution: 0.46

Log Likelihood =

Solution: −0.94

(d) Draw the same graph as in part (b), but using the new weights.

Solution:



(e) Perform one more update of gradient ascent, what is the new **w**? What are the new predictions and log likelihood (natural log)?

w =

Solution: [2.14, -0.28]

$P(y^{(1)}|\mathbf{x}^{(1)}; \mathbf{w})$ =

Solution: 0.86

$P(y^{(2)}|\mathbf{x}^{(2)}; \mathbf{w}) =$

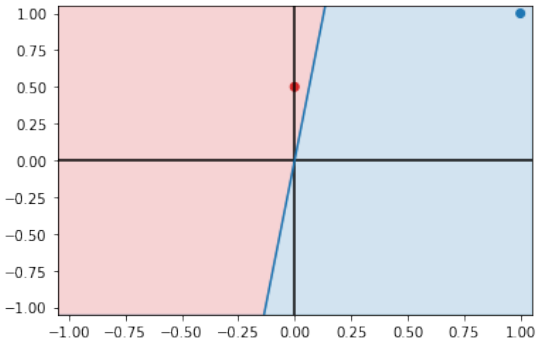
Solution: 0.53

Log Likelihood =

Solution: −0.77

(f) Draw the graph again with your new weights.

Solution:



(g) Briefly describe how gradient ascent moved the decision boundary, changed the log likelihood, and updated the classification probabilities during the course of training.

Solution: Gradient ascent causes the decision boundary to be moved in between the two datapoints, thus increasing the classification probabilities and also increasing the log likelihood throughout training.

3 Decision Trees [13 pts]

Covid cases are finally on a downward curve! As we all take measures to stay safe and masked-up whenever necessary, you've been collecting data from your friends and family to ensure their well-being.

Data collected includes covid symptoms as features and whether the person tested positive or not as the target. The symptoms are Headache, Fever and Cough where each attribute can have two values : Yes/No.
Given this information, answer 1 and 2 :

1. (1 point) How many distinct decision trees can be constructed?

   Solution: 256 (2^(2^3))
   3 boolean attributes

2. (1 point) How many decision stump models can be constructed?

   Solution: 12 (4*3)
   3 boolean attributes

   Now consider the samples shown below to answer 3-7

Headache	Fever	Cough	Tested Positive
No	Yes	No	No
Yes	No	Yes	Yes
No	No	No	No
No	Yes	Yes	Yes
Yes	Yes	Yes	Yes
Yes	No	No	Yes
Yes	Yes	No	No
No	No	Yes	No

3. (3 points) While constructing a decision tree to predict the target label, what is the information gain for each of the attributes assuming we're constructing the root node?
   (a) IG (Headache) =

   Solution:

Headache	Target=Yes	Target=No	Total
No	1	3	4
Yes	3	1	4

   IG (Headache) = I(0.5,0.5) - remainder(headache)
   remainder(headache) = (4/8)*I(1/4,3/4) + (4/8)*I(3/4,1/4) = 2*0.5* [(-1/4)log(1/4) + (-3/4)log(3/4)] = 0.8112
   IG (Headache) =1 - 0.8112 = 0.1888

   (b) IG (Fever) =

Solution:

Fever	Target=Yes	Target=No	Total
No	2	2	4
Yes	2	2	4

IG(Fever) = I(0.5,0.5) - remainder(Fever)
remainder(Fever) = (4/8)*I(2/4,2/4) + (4/8)*I(2/4,2/4)= 2*0.5* 2* (-1/2)log(1/2) =1
IG (Fever) =1-1 = 0

(c) IG (Cough) =

Solution:

Cough	Target=Yes	Target=No	Total
No	1	3	4
Yes	3	1	4

IG(Cough) = I(0.5,0.5) - remainder(Cough) = 0.1888

4. (1 point) Based on the information gain calculated above, which of the following attributes could be at the root of the resulting tree? Select all possible answers.

(i) Headache
(ii) Fever
(iii) Cough

Solution: (i) Headache and (ii) Cough
(maximal information gain at the root)

5. (1 point) In the final decision tree, how many edges does the longest path have (following the standard algorithm in Lec9 slide 47, with any arbitrary function to choose the split feature)? Select all possible answers.

(i) 1
(ii) 2
(iii) 3
(iv) 4

Solution: (iii) 3. Regardless of the choice of the feature at the root, the resulting tree needs to consider all 3 features in a path, so there are 3 edges in that path.
Split on Headache at the root.
For Headache = Yes :

Fever	Target=Yes	Target=No	Total
No	2	0	2
Yes	1	1	2

Cough	Target=Yes	Target=No	Total
No	1	2	2
Yes	2	0	2

Suppose the node is split on Fever, for Headache=Yes and Fever=Yes, we still need to know Cough value to decide the final target. Similarly, if the node is split on Cough, Headache=Yes and Cough=No are not

sufficient to decide the final target (Fever value is needed).
Thus the longest path will need to consider all the features.

6. (1 point) State whether True or False : If a person has both fever and cough, we cannot always say they will test positive for Covid **(solely based on the final decision tree)**.

Solution: False. Fever=Yes and Cough=Yes always results in TestedPositive = Yes.
No matter the way we split these features, all of the training data samples with Fever=Yes and Cough=Yes point to TestedPositive=Yes.

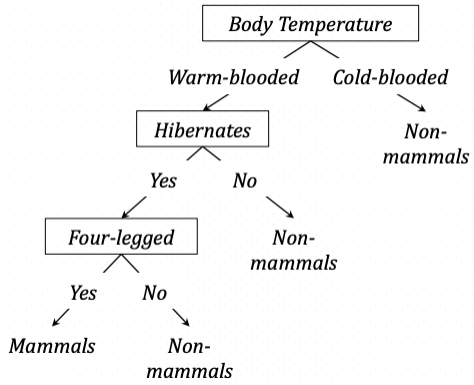
7. (1 point) An additional feature "Temperature" is added to the dataset. It contains numerical values denoting the body temperature of the person in Fahrenheit.
   (a) Is it still possible to construct a decision tree on the new dataset? (Yes / No)
   (b) If No, explain why. If Yes, how would the data be split?

Solution: (a) Yes
(b) Since temp is a continuous feature, split on the median value of the all the examples.

8. (4 points) The below data includes different features of animals to classify whether they are mammals or not.

Name	Body Temperature	Gives Birth	Four-legged	Hibernates	Class Label
Salamander	Cold-blooded	No	Yes	Yes	***No***
Guppy	Cold-blooded	Yes	No	No	***No***
Eagle	Warm-blooded	No	No	No	***No***
Poorwill	Warm-blooded	No	No	Yes	***No***
Platypus	Warm-blooded	No	Yes	Yes	***Yes***

From this data, the following decision tree is constructed :



(a) What would the above tree classify humans as? (We are four-legged warm-blooded animals that can give birth and don't hibernate)

Solution: Non-mammals

(b) What is the reason for the above classification?

Solution: Overfitting

(c) Briefly describe why the above tree has overfit the dataset.

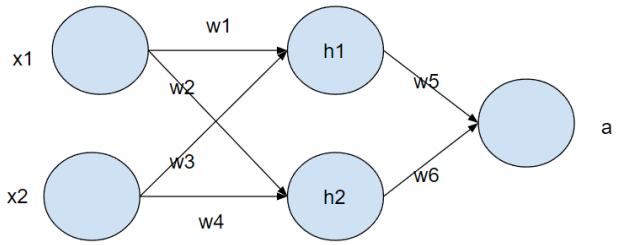
Solution: There are lack of representative instances to model the tree on.
Incorrect answer : noise (there aren't noisy samples in the dataset)

(d) Describe any one way to reduce overfitting in decision trees.

Solution: Pruning (pre-pruning, post-pruning, limit the hypothesis space, limit max-depth of trees) / Cross validation / Chance cut-off

4 Neural Networks [14 pts]

1. (10 points) We will be looking at the *backpropagation* algorithm to calculate the optimal values of the weights. It allows for efficiently computing the gradients of neural network hidden layer weights with respect to the objective function.
   Consider the neural network below with 2 input nodes, two neurons in the hidden layer and one output. Hidden layer has ReLU activation functions and the output has a sigmoid activation function. This network can be used for classification. Since the final output is a probability value between 0 and 1, values >=0.5 will be assigned label 1 and values <0.5 will be assigned label 0. We will use the example below to show how backprop works:



   Recall the gradient of the log of the likelihood function we calculated in Q2.2 for logistic regression:

$$\begin{aligned}\nabla_{\mathbf{w}} \log \mathcal{L}(w) &= \nabla_{\mathbf{w}} \sum_{i=1}^{N} y^{(i)} \log \phi(\mathbf{w} \cdot \mathbf{x}^{(i)}) + (1 - y^{(i)}) \log(1 - \phi(\mathbf{w} \cdot \mathbf{x}^{(i)})) \\ &= \sum_{i=1}^{N} \left[y^{(i)} - \phi(\mathbf{w} \cdot \mathbf{x}^{(i)}) \right] \mathbf{x}^{(i)} \end{aligned} \tag{2}$$

   Notice that the last layer of this neural network, represented by w_5 and w_6, is essentially performing logistic regression over the input features h_1 and h_2. We can represent w_5 and w_6 with a weight vector $\mathbf{w} = \begin{bmatrix} w_5 \\ w_6 \end{bmatrix}$, and inputs h_1 and h_2 with a feature vector $\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}$. For this neural network with output a and label y, the derivative of the max likelihood function with respect to w_5 and w_6 is then the first and second components of

$$\sum_{i=1}^{N} \left[y^{(i)} - \phi(\mathbf{w} \cdot \mathbf{h}^{(i)}) \right] \mathbf{h}^{(i)}$$

   Written more clearly,

$$\begin{aligned}\frac{\partial}{\partial w_5} \log \mathcal{L}(w_1, ..., w_6) &= \left[\sum_{i=1}^{N} \left[y^{(i)} - \phi(\mathbf{w} \cdot \mathbf{h}^{(i)}) \right] \mathbf{h}^{(i)} \right]_1 \quad \text{where the subscript denotes the first element of the vector} \\ &= \sum_{i=1}^{N} \left[y^{(i)} - \phi(\mathbf{w} \cdot \mathbf{h}^{(i)}) \right] h_1^{(i)} \\ &= \sum_{i=1}^{N} \left[y^{(i)} - \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)}) \right] h_1^{(i)} \end{aligned}$$

   Thus, we can update w_5 through gradient ascent to maximize the log likelihood of the data by updating

$$w_5 \leftarrow w_5 + \alpha \left[y - \phi(w_5 h_1 + w_6 h_2) \right] h_1$$

(a) Derive $\frac{\partial}{\partial w_6} \log \mathcal{L}(w_1, ..., w_6)$ as a function of $y^{(i)}, w_5, w_6, h_1^{(i)}, h_2^{(i)}$.

Solution: $\frac{\partial}{\partial w_6} \log \mathcal{L}(w_1, ..., w_6) = \sum_{i=1}^{N} \left[y^{(i)} - \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)}) \right] h_2^{(i)}$

(b) Now, let's consider how to update the weights in the *first* layer. Recall the chain rule: for functions f and g, $\frac{\partial}{\partial x} f(g(x)) = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}$.

As an example, consider $f(x) = w_1 * g(x)$, $g(x) = w_2 * x$. Then, $\frac{\partial f(g(x))}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x} = w_1 * w_2$.

In the first layer, we have input variables x_1 and x_2. We can represent w_1, w_2, w_3 and w_4 with a weight matrix $\mathbf{W} = \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \end{bmatrix}$, and inputs x_1 and x_2 with a vector $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Then the hidden layer values are the first and second components of $ReLU(\mathbf{W}\mathbf{x}^{(i)})$, i.e.

$$h_1^{(i)} = [ReLU(\mathbf{W}\mathbf{x}^{(i)})]_1$$
$$h_2^{(i)} = [ReLU(\mathbf{W}\mathbf{x}^{(i)})]_2$$

Find the derivative of the log likelihood L with respect to w_1, using the chain rule for differentiation provided to you above. (First differentiate with respect to the ReLU function (see class slides) in h and then with respect to w_1). Here is a step by step breakdown to help you.

(i) First, write $\frac{\partial}{w_1} \log \mathcal{L}(w_1, ..., w_6)$ in terms of the product of two partial derivatives. The first term should be a partial with respect to one of the h and the second should be a partial with respect to w_1.

(ii) Now, compute the first partial. Use the gradient derivation given above for ($\nabla_w \log \mathcal{L}(w)$, Eq. 2) and notice some symmetry to do this part more easily.

(iii) Note that the ReLU function has two cases. Take the derivative of each case separately.

(iv) Now, put it all together using the chain rule from the first step. You will have two cases for the final gradient.

(v) You now know how to update the weights of a 2-layer ReLU neural network for binary classification! This derivation forms the backbone of backpropagation, which is used to efficiently update neural network parameters on basically all neural networks deployed right now (Tesla's self-driving cars, Siri voice understanding, Google's text autocomplete, etc.).

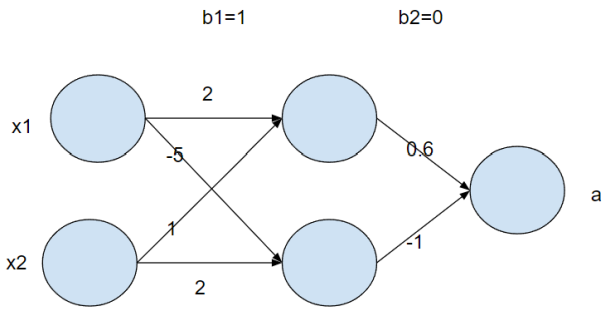
Solution: $\frac{\partial}{\partial w_1} \log \mathcal{L}(w_1, ..., w_6) = \frac{\partial \mathcal{L}}{\partial h_1^{(i)}} * \frac{\partial h_1^{(i)}}{\partial w_1}$

$$\frac{\partial \mathcal{L}}{\partial h_1} = \frac{\partial}{\partial h_1} \sum_{i=1}^{N} y^{(i)} \log \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)}) + (1 - y^{(i)}) \log(1 - \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)}))$$
$$= \sum_{i=1}^{N} [y^{(i)} - \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)})] * w_5$$

$h_1^{(i)} = ReLU(w_1 x_1^{(i)} + w_3 x_2^{(i)})$

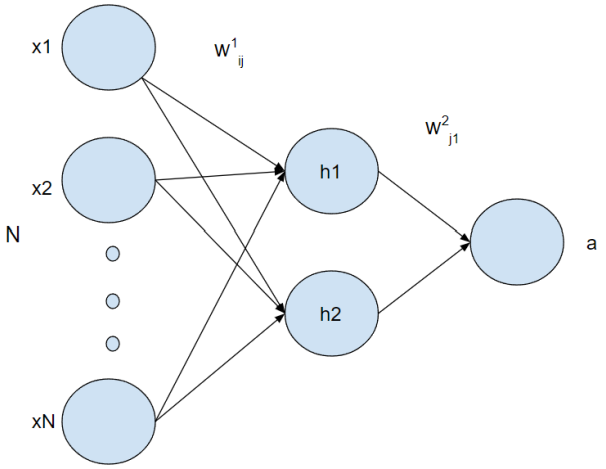
$$\frac{\partial}{\partial w_1} h_1^{(i)} = \begin{cases} x_1^{(i)} & \text{if } h_1^{(i)} >= 0 \\ 0 & \text{otherwise} \end{cases}$$
$$\implies \frac{\partial}{\partial w_1} L = \begin{cases} \sum_{i=1}^{N} [y^{(i)} - \phi(w_5 h_1^{(i)} + w_6 h_2^{(i)})] * w_5 x_1^{(i)} & \text{if } h_1^{(i)} >= 0 \\ 0 & \text{otherwise} \end{cases}$$

(c) Considering $N = 2$ (number of input variables), perform forward propagation using the values for biases and weights given in the figure. Consider $x1 = 2$ and $x2 = 1$ and calculate the numerical value of the output a. Don't forget to add the biases:



Solution: 0.973

(d) Consider the same 2-layer neural network given initially but with N input nodes instead of just two inputs. Answer the questions below given only a single training example $X = [x_1, x_2, ...x_N]$, weights w^1_{ij}, w^2_{j1}, output a and actual label y. (Note that i takes values from 1 to N, i.e. input size, whereas j takes the values 1,2 only i.e., the number of neurons in hidden layer).



Derive an expression for the output a in terms of the inputs and weights given in the initial figure, i.e, w^1_{ij}, w^2_{j1} and x_i. Use ϕ to denote the sigmoid function. Clearly show the value of each hidden neuron and perform forward propagation step-by-step.

Solution: $input : x_1, x_2, ...x_N$
weight 1 : w^1_{ij}
weight 2 : w^2_{j1}
Input to the hidden layer = $\sum_{i=1}^{N} w^1_{ij} x_i$
$h_j = ReLU(\sum_{i=1}^{N} w^1_{ij} x_i)$
This hidden layer value is the input to the output layer
$a = \phi(\sum_{j=1}^{2} w^2_{j1} h_j)$
$a = \phi(\sum_{j=1}^{2} w^2_{j1} * ReLU(\sum_{i=1}^{N} w^1_{ij} x_i))$

(e) Derive the equation for the weight update for the weight matrix entries w^1_{ij} from (d). Assume that the actual label for $[x_1, x_2, .., x_N]$ is y and that we are using maximum likelihood estimation. Follow a similar procedure as to part (b) by using the chain rule to derive the gradient with respect to w^1_{ij}.

Solution: First differentiate wrt h_j and then wrt w^1_{ij} to find the gradient wrt w^1_{ij}

$\frac{\partial}{\partial w^1_{ij}} L = \frac{\partial}{\partial h_j} L * \frac{\partial}{\partial h_j} w^1_{ij}$

$\frac{\partial}{\partial h_j} L = [y - \phi(\sum_{j=1}^{2} w^2_{j1} h_j)] w^2_{j1}$

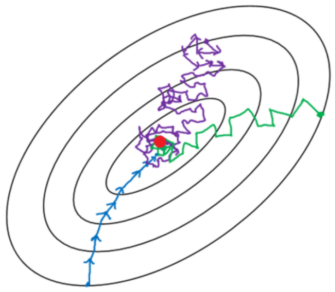
$$\frac{\partial}{\partial h_j} w^1_{ij} = \begin{cases} x_i & \text{if } h_j >= 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\frac{\partial}{\partial w^1_{ij}} L = \begin{cases} [\text{y - } \phi(\sum_{j=1}^{2} w^2_{j1} h_j)] w^2_{j1} x_i & \text{if } h_j >= 0 \\ 0 & \text{otherwise} \end{cases}$$

2. (2 points) Just as we use a coordinate plane to represent a graph with two variables (X and Y), we can represent a function with three variables on a two-dimensional plane using a contour plot. The graph will contain a number of contour lines with each line having a constant value on a third variable. Similar to a coordinate plane, the vertical and horizontal axis represent, individually, an independent variable. It is commonly used to show how a value Z changes as a function of two variables, X and Y.

   Below is a contour plot showing three different gradient ascent algorithms being run until they reach the same maxima: batch gradient ascent, mini-batch gradient ascent and stochastic gradient ascent.

   Indicate which color - blue, green, purple - corresponds to which form of gradient ascent, briefly explain *why* for each choice. Assume that there are N training points, a batch size of B for mini-batch gradient ascent, and $1 << B << N$.



Solution:

- Blue - Batch Gradient Descent. This is the least noisy line as we are taking gradient steps computed over the full dataset.
- Green - Mini-Batch Gradient Descent. This is somewhat noisy as we are using batches to compute the gradient.
- Purple - Stochastic Gradient Descent. This is very noisy as we are taking one step at a time.

3. (2 points) In class we learned about the tradeoffs in bias and variance in different ML models. How would the bias and variance of a neural network change when changing the following hyperparameters of the network? Briefly explain why.

   (a) Increasing the number of layers and neurons in each layer.

Solution: Decreases bias and increases variance. More layers and neurons in each layer decreases training error and fits the training data better, decreasing bias and consequently increasing variance.

(b) Increasing number of examples in the training set.

Solution: Decreases variance. Exposing the model to more training samples helps lower the variance as it helps reduce overfitting.

(c) Adding regularization by limiting the values of the weights.

Solution: Increases bias and decreases variance. Regularization reduces the variance of the model by simplifying it and consequently increasing bias.

5 Bayes Nets [14 pts]

1. (2 points) Given two 6-sided dice D1 and D2 to roll, answer the following questions.
   (a) Let E be the event where sum of the values on D1 and D2 sum up to 5. What is the probability of occurrence of event E, i.e P(E)

   Solution: P(E) = 4/36 = 1/9 {(2, 3) (3, 2) (1, 4) (4, 1)}

   (b) If F is an event of observing value on D2 to be 4, what is the probability of occurrence of event E given F, i.e. P(E | F)

   Solution: 1/36

2. (2 points) Let X, Y be two independent random variables with domain {1, 2, 3}. Given the below few values for a joint distribution and the value P(Y=3) = 1/2, fill in the remaining values.

P(X=1, Y=1)=3/32	P(X=1, Y=2)=1/32	P(X=1, Y=3)= ___
P(X=2, Y=1)=3/48	P(X=2, Y=2)=1/48	P(X=2, Y=3)= ___
P(X=3, Y=1)= ___	P(X=3, Y=2)= ___	P(X=3, Y=3)= ___

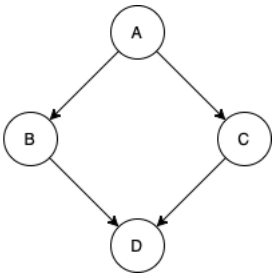
   Solution: Since, the given variables are independent, we know $P(X, Y) = P(X)(Y)$.
   $P(X = 1, Y = 1)/P(X = 1, Y = 2) = P(Y = 1)/P(Y = 2) = 3$
   $P(X = 1, Y = 1)/P(X = 2, Y = 1) = P(X = 1)/(X = 2) = 3/2$
   $P(Y = 1) + P(Y = 2) + P(Y = 3) = 1$
   $4P(Y = 2) + 1/2 = 1$
   $P(Y = 2) = 1/8$
   $P(Y = 1) = 3/8$
   $P(X = 1) = 3/32 * 1/P(Y = 1) = 1/4$
   $P(X = 2) = 3/48 * 1/P(Y = 1) = 1/6$
   $P(X = 3) = 1 - 1/4 - 1/6 = 7/12$

   $P(X = 1, Y = 3) = 1/8$, $P(X = 2, Y = 3) = 1/12$,
   $P(X = 3, Y = 1) = 7/32$, $P(X = 3, Y = 2) = 7/96$,
   $P(X = 3, Y = 3) = 7/24$

3. (2 points) Suppose that there is a 1 in 100 chance that a plane crashes. Given the probability that a pilot is sleep-deprived when the plane crashes is 0.7 and the probability they're sleep-deprived is 0.1 when it doesn't crash. if the pilot is sleep-deprived, what is the probability that the plane crashes?

   Solution: 0.039
   Let the event that pilot crashes be represented by C and the event pilot is sleep deprived be S, We are given the below values:
   $P(+c) = 0.01$
   $P(+s| + c) = 0.7$
   $P(+s| - c) = 0.1$
   $P(+c| + s) = P(+s| + c)P(+c)/P(+s)$
   We know, $P(+s) = P(+s| + c) * P(+c) + P(+s| - c) * P(-c)$
   $P(+s) = 0.176$
   $P(+c| + s) = (0.7 * 0.01)/(0.176) = 0.039$

4. (2 points) Given the following Bayesian Network with variables A, B, C, D that are of boolean values, What is the probability of D being true given A is true?



A	P(A)
+	0.25
-	0.75

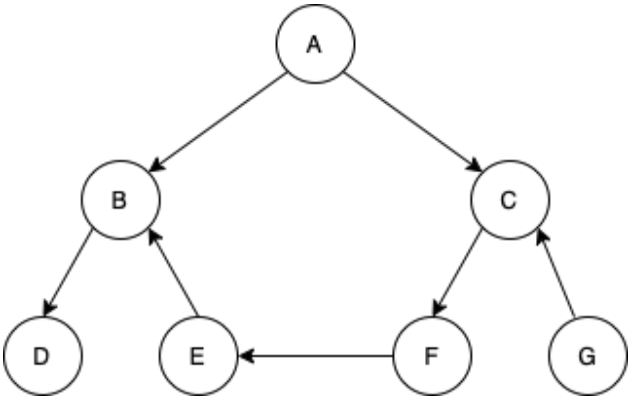
B	A	P(B | A)
+	+	0.1
-	+	0.9
+	-	0.4
-	-	0.6

C	A	P(C | A)
+	+	0.2
-	+	0.8
+	-	0.3
-	-	0.7

D	B	C	P(D | B, C)
+	+	+	0.1
-	+	+	0.9
+	+	-	0.25
-	+	-	0.75
+	-	+	0.3
-	-	+	0.7
+	-	-	0.2
-	-	-	0.8

Solution: 0.22
$P(+d|+a) = P(+d,+a)/P(+a)$
$P(+d,+a) = \sum_b \sum_c P(+d,b,c,+a) = P(+d,+a,+b,+c)+P(+d,+a,+b,-c)+P(+d,+a,-b,+c)+P(+d,+a,-b,-c)$
$P(+d,+a,+b,+c) = P(+a) * P(+b|+a) * P(+c|+a) * P(+d|+b,+c) = 0.25 * 0.1 * 0.2 * 0.1$
Similarly, $P(+d,+a,+b,-c) = 0.25 * 0.1 * 0.8 * 0.25$,
$P(+d,+a,-b,+c) = 0.25 * 0.9 * 0.2 * 0.3$,
$P(+d,+a,-b,-c) = 0.25 * 0.9 * 0.8 * 0.2$
$P(+d,+a) = 0.25 * 0.22$
$P(+d|+a) = P(+d,+a)/P(+a) = 0.22$

5. (2 points) Given the below Bayesian Network, which of the following assumptions of (conditional) independence are valid?



(a) $D \perp\!\!\!\perp A|B$

Solution: true

(b) $A \perp\!\!\!\perp G$

Solution: true

(c) $A \perp\!\!\!\perp G|F$

Solution: false

(d) $A \perp\!\!\!\perp E|B, F$

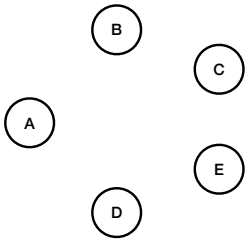
Solution: false

6. (4 points) You're performing variable elimination for HW2 on a Bayes Net with 5 variables: A, B, C, D, E. Suddenly, your dog comes along and eats your homework! You have now lost the original structure of the Bayes net, but fortunately, you remember where you left off. You have just finished joining over the variable C, and but you have not yet eliminated C nor joined over any other variable.

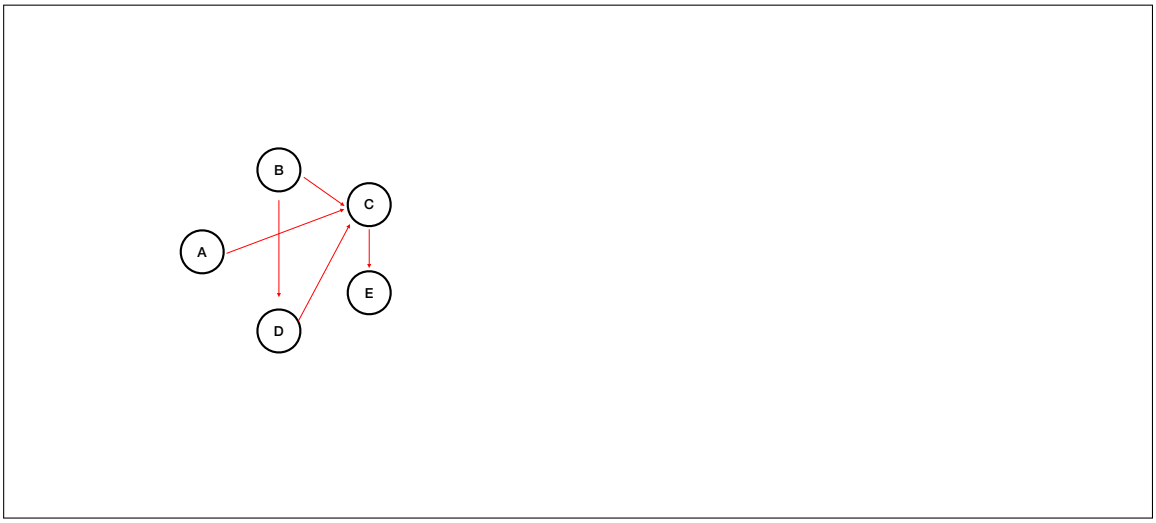
   You currently have four factors (meaning, either normalized or unnormalized probabilities) so far, the first one involving just (A), the second involving just (B), the third involving (B, D), and the fourth involving (A, B, C, D, E). In each factor, you don't remember which variables are conditioned on, or if at all.

   HINT: When you are joining over C, you know that there must be at least one term within the factor that looks like $P(*|C, *)$, and another like $P(C, *|*)$ (otherwise we won't be able to sum C out). Use this fact and what's in the other factors, to figure out which nodes MUST have certain parents and which nodes CANNOT.

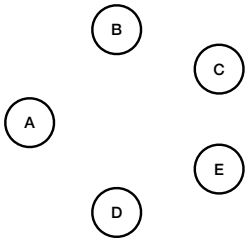
   (a) What's the smallest number of edges possible in the original Bayes Net? List it out and then draw an example.



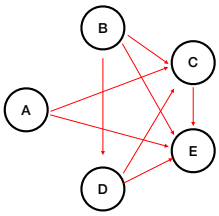
Solution: 5. Since we haven't summed over any variable yet, each factor tells which variables must have edges with each other. There's a factor involving just A and one involving just B so they must not have any parents (the two factors are $P(A)$ and $P(B)$). The one involving (B, D) then must have B as a parent of D since B has no parents. In the last one, since we're joining over C, the factor must have a term like $P(*|C)$, and another one like $P(C, *|*)$ (otherwise the join and summation wouldn't get rid of C). Since nothing else involves E, then C must be a parent of E for this to be true so the factor could be $P(E|C)$. Finally, this last factor involves A, B, D. D can't have any other parent than B otherwise the third factor (B, D) would have to have another variable in it. So then D must point to C or E. A and B both can't have more parents, so to be included in this factor they can point to either C or E.



(b) What's the largest number of edges possible in the original Bayes Net? List it out and then draw an example.



Solution: 8. Similar to the logic from above, we know that A and B must be parents, and that D depends on B. The last factor can change to have more edges though. The factor must have a term like $P(*|C)$, along with other possible terms involving A,B,C,D,and E. We know that D can't have any more parents, because if it did then the factor involving (B, D) would also have another variable for D's parent. Therefore, just like before, C must be a parent of E, but neither C nor E can be parents of D. However, we can make A, B, D parents of both C/E since this doesn't violate any constraints.



6 Perceptrons [14 pts]

1. (4 points) **Answer the following questions**

   (a) Is a single perceptron is capable of computing an XNOR function?
   i. True
   ii.False

   Solution: False

   (b) Is the perceptron algorithm guaranteed to converge within a finite number of training steps for a linearly separable dataset?
   i. True
   ii.False

   Solution: True

   (c) Is the perceptron algorithm guaranteed to find a max margin separating hyperplane for a linearly separable dataset?
   i. True
   ii.False

   Solution: False

   (d) Will the results of a perceptron algorithm will always converge? Give a reason for your answer.

   Solution: A. False – When the data is not linearly separable.

   (e) If we run the perceptron algorithm with no bias on 'd' dimensional data, the decision boundary produced will be a hyperplane passing through the origin R^d, even when it doesn't converge.
   i. True
   ii.False

   Solution: True

   (f) Steve wants to use a perceptron for his binary data $x^{(i)}$ with labels $y^{(i)} \in \{+1, -1\}$. Steve observed that his data was not linearly separable and hence decided to adopt an approach where he would train with data $z^{(i)} = (x^{(i)}, y^{(i)})$ and labels as $y^{(i)}$. Is this new approach successful in changing it into a linearly separable dataset? Why or why not?

   Solution: Yes, because there is a new dimension in which all points in one class are above the origin and all points in the other class are below the origin (using the +1, -1 labels). A separating hyperplane exists by directly cutting through the origin with last dimension = 0.

   (g) Suppose we have a linearly separable four-class dataset with classes and we run the perceptron algorithm, with initial weights $\mathbf{w}_1^0, \mathbf{w}_2^0, \mathbf{w}_3^0, \mathbf{w}_4^0$ (superscript denotes the number of updates, subscript denotes each class' vector), until convergence. Let $|D|$ be the number of data points and t be the number of updates to the weights before convergence. If $x = \mathbf{w}_1^0 + \mathbf{w}_2^0 + \mathbf{w}_3^0 + \mathbf{w}_4^0$ then what is the sum of $\mathbf{w}_1^t + \mathbf{w}_2^t + \mathbf{w}_3^t + \mathbf{w}_4^t$?

   Solution: x. Each time we classify incorrectly, we add $f(x)$ to one weight subtract $f(x)$ from another. If we classify correctly, we make no change to the weights.

2. (4 points) **Multi-Class Perceptron**

(a) Suppose we have a multiclass perceptron with classes A, B, C and initial weights set to $\mathbf{w}_A = [1, 3]^T$, $\mathbf{w}_B = [2, -1]^T$, and $\mathbf{w}_C = [-3, -2]^T$. Write the resulting weight vectors after training on the following datapoint.

X0	X1	Label
2	2	A

$\mathbf{w}_A$ = ?
$\mathbf{w}_B$ = ?
$\mathbf{w}_C$= ?

Solution: The predicted label is $argmax_{y\in(A,B,C)}\mathbf{w}_y^T\mathbf{x}$ which is A.
$\mathbf{w}_a^T\mathbf{x} = 1 * 2 + 3 * 2 = 8$ is greater than both $\mathbf{w}_B^T\mathbf{x} = 2 * 2 - 1 * 2 = 2$ and $\mathbf{w}_C^T\mathbf{x} = -3 * 2 - 2 * 2 = -10$. Therefore the weights are not updated, so the weights are the same as the initial ones.
$\mathbf{w}_A = [1, 3]^T$
$\mathbf{w}_B = [2, -1]^T$
$\mathbf{w}_C = [-3, -2]^T$

(b) Suppose we have a different multiclass perceptron with classes A, B, C and initial weights set to Wa = [1, 3], Wb = [2,-1] and Wc = [-3,-2]. Write the vectors Wa, Wb, Wc after training on the following 2-dimensional training data once.

X0	X1	Label
-5	2	B

$\mathbf{w}_A$ = ?
$\mathbf{w}_B$ = ?
$\mathbf{w}_C$= ?

Solution: The predicted label is C, since since $\mathbf{w}_C^T\mathbf{x} = 11$ is greater than both $\mathbf{w}_A^T\mathbf{x} = 1, \mathbf{w}_B^T\mathbf{x} = -12$. Since the predicted label C is not equal to the correct label B, the weights are updated by subtracting the features from the weights of the predicted label and by adding to the weights of the correct labels.
$\mathbf{w}_C = \mathbf{w}_C - [-5, 2]^T = [2, -4]^T$
$\mathbf{w}_B = \mathbf{w}_B + [-5, 2]^T = [-3, 1]^T$
$\mathbf{w}_A = [1, 3]^T$

3. (3 points) **Perceptron Learning Algorithm** The table shown below displays points in the distribution $\mathbb{R}^2$. Suppose we run the perceptron algorithm. We then note the total number of times each point participates in the gradient descent step since it is misclassified, throughout the run.

X1	X2	Y	Times Misclassified
-3	2	+1	2
-1	1	+1	1
1	-1	-1	1
-1	-1	-1	0
2	2	-1	0

(a) Suppose that the learning rate is $\alpha = 0.8$, and the initial weight vector is $\mathbf{w}^0 = [-3, 2, 1]^T$, where the bias term is the last component. What is the equation of the separating line in terms of the features x_1 and x_2 found by the perceptron algorithm?

> **Solution:** At each iteration, the weights are updated by picking a misclassified point and applying the update rule. The learned weights are $w = w^0 + \alpha * \sum(\epsilon^{(i)} * y^{(i)} * x^{(i)})$, where the variable $\epsilon^{(i)}$is the number of times the i'th point is misclassified. Recall that we augment each point x with $x_3 = 1$ for the bias. Thus, we have $w = (-3, 2, 1) + 0.8(2 * 1 * (-3, 2, 1) + 1 * 1 * (-1, 1, 1) + 1 * -1 * (1, -1, 1)) = (-9.4, 6.8, 2.6)$. Therefore, the equation of the separating line is $-9.4x_1 + 6.8x_2 + 2.6 = 0$.

(b) For which, if any, point or points in our dataset would the learned decision boundary change if we removed it? Explain your answer.

> **Solution:** If we removed either of the 3 points that were misclassified during training, it would cause a change in the learned decision boundary.

(c) How would our result differ if we were to add the additional training point (-1, -1) with label +1?

> **Solution:** The data would no longer be linearly separable, so the perceptron algorithm would not terminate.

4. (3 points) **Perceptron Update Rule as Stochastic Gradient Ascent**: Recall the binary perceptron update rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \begin{cases} 0 & \text{if } y = y^* \\ y^*\mathbf{x} & \text{if } y \neq y^* \end{cases}$$

where y^* is the true class label, and $y \in \{-1, 1\}$ is the binary class prediction produced by the perceptron on datapoint x. Notice that gradient ascent's update rule is also of a similar form, but it is $\mathbf{w} \leftarrow \mathbf{w} + \nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ for some objective function $\mathcal{L}(\mathbf{w})$.

We can thus recast the perceptron update rule as a form of stochastic (single-sample, but without randomness) gradient ascent on some unknown objective function. **Find the objective function that we are *maximizing* given by the perceptron update rule** (disregarding any constants that don't change the parameter that maximizes the function), and **briefly explain what it intuitively means**. It should be expressed as a single term, without cases, as a function of $y^*, \mathbf{w}, \mathbf{x}$. Hints: 1) the objective involves a min over two terms, and think about the sign of $\mathbf{w} \cdot x$; 2) $\int \mathbf{x}\, d\mathbf{w} = \mathbf{w} \cdot \mathbf{x} + C$ when $\mathbf{w}, \mathbf{x}$ are vectors (C is a constant vector from integrating).

Objective: $\max_{\mathbf{w}} \min(\quad , \quad)$

> **Solution:** Integrate both cases. If $y = y^*$, then the integral of 0 is C where C is an integration constant that we will ignore. If $y \neq y^*$, then the integral of $y^*\mathbf{x}$ is $y^*\mathbf{w} \cdot \mathbf{x} + C$ and we can disregard the constant C. Combining these two, we have that the objective function is $\max_{\mathbf{w}} \min(0, y^*\mathbf{w} \cdot \mathbf{x})$.
>
> This means that the objective is to maximize classification accuracy as measured by 0 if the prediction is correct, and a negative value if the prediction is incorrect.