CS221 Midterm Solutions

CS221
November 18, 2014

Name: _____
by writing my name I agree to abide by the honor code

SUNet ID: _____

Read all of the following information before starting the exam:

- This test has 3 problems and is worth 150 points total. It is your responsibility to make sure that you have all of the pages.
- Keep your answers precise and concise. Show all work, clearly and in order, or else points will be deducted, even if your final answer is correct.
- Don't spend too much time on one problem. Read through all the problems carefully and do the easy ones first. Try to understand the problems intuitively; it really helps to draw a picture.
- Good luck!

Problem	Part	Max Score	Score
1	a	10	
	b	10	
	c	10	
	d	10	
	e	10	
2	a	10	
	b	10	
	c	10	
	d	10	
	e	10	
3	a	10	
	b	10	
	c	10	
	d	10	
	e	10	

Total Score: ☐ + ☐ + ☐ = ☐

1. Enchaining Realm (*50 points*)

This problem is about machine learning.

a. (*10 points*)

Suppose we want to predict a real-valued output $y \in \mathbb{R}$ given an input $x = (x_1, x_2) \in \mathbb{R}^2$, which is represented by a feature vector $\phi(x) = (x_1, |x_1 - x_2|)$.

Consider the following training set of (x, y) pairs:

$$\mathcal{D}_{\text{train}} = \{((1,2),2), ((1,1),1), ((2,1),3)\}. \tag{1}$$

We use a modified squared loss function, which penalizes overshooting twice as much as undershooting:

$$\text{Loss}(x, y, \mathbf{w}) = \begin{cases} \frac{1}{2}(\mathbf{w} \cdot \phi(x) - y)^2 & \text{if } \mathbf{w} \cdot \phi(x) < y \\ (\mathbf{w} \cdot \phi(x) - y)^2 & \text{otherwise} \end{cases} \tag{2}$$

Using a fixed learning rate of $\eta = 1$, apply the stochastic gradient descent algorithm on this training set starting from $\mathbf{w} = [0, 0]$ after looping through each example (x, y) in order and performing the following update:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w}). \tag{3}$$

For each example in the training set, calculate the loss on that example and update the weight vector $\mathbf{w}$ to fill in the table below:

	x	$\phi(x)$	$\text{Loss}(x, y, \mathbf{w})$	$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$	weights $\mathbf{w}$
Initialization	n/a	n/a	n/a	n/a	$[0, 0]$
After example 1	(1,2)				
After example 2	(1,1)				
After example 3	(2,1)				

Solution

	x	$\phi(x)$	$\text{Loss}(x, y, \mathbf{w})$	$\nabla_{\mathbf{w}} \text{Loss}(x, y, \mathbf{w})$	weights $\mathbf{w}$
Initialization	n/a	n/a	n/a	n/a	$[0, 0]$
After example 1	(1,2)	(1,1)	2	$[-2, -2]$	$[2, 2]$
After example 2	(1,1)	(1,0)	1	$[2, 0]$	$[0, 2]$
After example 3	(2,1)	(2,1)	0.5	$[-2, -1]$	$[2, 3]$

b. (*10 points*)

Consider the following set of 6 points in the plane:

$$\{A = (0, 0), B = (1, 0), C = (0, 3), D = (1, 3), E = (5, 7), F = (8, 12)\} \tag{4}$$

You'd like to partition the points into two clusters, where each cluster is represented by a centroid μ_k that is *constrained to lie on the diagonal line*; formally, $\mu_k = [c_k, c_k]$ for $k \in \{1, 2\}$.

Recall that the reconstruction loss is the sum of squared distances from each point to the centroid it is assigned to. Modify the K-means algorithm to minimize this loss while respecting the diagonal constraint.

Suppose we have $K = 2$ clusters which are initialized with $c_1 = 3$ and $c_2 = 11$. Break ties, if any, by assigning a point to the centroid farther away from the origin. Run two iterations of K-means, filling out the values below:

Iteration	**Task**	**Value**
1.	Assignment $z_i = 1$:	(i) _____
1.	Assignment $z_i = 2$:	(ii) _____
1.	New c_1:	(iii) _____
1.	New c_2:	(iv) _____
1.	Reconstruction loss after updating centroids:	(v) _____
2.	Assignment $z_i = 1$:	(vi) _____
2.	Assignment $z_i = 2$:	(vii) _____
2.	New c_1:	(viii) _____
2.	New c_2:	(ix) _____
2.	Reconstruction loss after updating centroids:	(x) _____

Solution

Iteration	Task	Value
1.	Assignment $z_i = 1$:	(i) A, B, C, D, E
1.	Assignment $z_i = 2$:	(ii) F
1.	New c_1:	(iii) 2
1.	New c_2:	(iv) 10
1.	Reconstruction loss:	(v) 62
2.	Assignment $z_i = 1$:	(vi) A, B, C, D
2.	Assignment $z_i = 2$:	(vii) E, F
2.	New c_1:	(viii) 1
2.	New c_2:	(ix) 8
2.	Reconstruction loss:	(x) 38

c. (*10 points*)

Given two images x_1 and x_2, our goal is to predict whether they are of the same person ($y = 1$) or not ($y = -1$).

We build the following model: We have a feature extractor that maps an image x to a feature vector $\phi(x) \in \mathbb{R}^d$. For each $j = 1, \ldots, K$, define $h_j(x) = \mathbf{v}_j \cdot \phi(x)$. Intuitively, each parameter $\mathbf{v}_j \in \mathbb{R}^d$ corresponds to a direction; $h_j(x)$ corresponds to the projection of $\phi(x)$ along that direction.

Given a weight vector $\mathbf{w} = (w_1, \ldots, w_K)$, the classification score on an input (x_1, x_2) is defined as:

$$s(x_1, x_2) = \sum_{j=1}^{K} w_j h_j(x_1) h_j(x_2). \tag{5}$$

As an example, for $K = d = 2$, if the parameters were $\mathbf{v}_1 = (1, 0), \mathbf{v}_2 = (0, 1), \mathbf{w} = (3, 4)$, then the classification score would be $3\phi(x_1)_1\phi(x_2)_1 + 4\phi(x_1)_2\phi(x_2)_2$.

Prove that there exists a new feature extractor A mapping (x_1, x_2) to a d^2-dimensional feature vector $A(x_1, x_2) \in \mathbb{R}^{d^2}$, such that for any scoring function s (defined by $\mathbf{v}_1, \ldots, \mathbf{v}_K, \mathbf{w}$), there exists a new weight vector $\mathbf{u} \in \mathbb{R}^{d^2}$ with $s(x_1, x_2) = \mathbf{u} \cdot A(x_1, x_2)$. In other words, the scoring functions in (**??**) can be represented by linear functions with appropriate features. You must define $\mathbf{u}$ explicitly.

Solution We will index the components of A and u by (i, j) (of which there are d^2 values); technically, if we want the indices to be $1, \ldots, d^2$, then we can index the components with $d(i-1)+j$. Define the components of $A(x_1, x_2)$ to be $\phi(x_1)_i\phi(x_2)_j$ for $1 \le i, j \le d$. Then for each component (i, j), set the weight $u_{i,j} = \sum_{k=1}^{K} w_k v_{k,i} v_{k,j}$. One can check via some algebra that this choice of $\mathbf{u}$ realizes the function $s(x_1, x_2)$.

d. (*10 points*)

Suppose we are performing classification where the input points are of the form $(x_1, x_2) \in \mathbb{R}^2$. We can choose any subset of the following set of features:

$$\mathcal{F} = \left\{ x_1^2, x_2^2, x_1 x_2, x_1, x_2, \frac{1}{x_1}, \frac{1}{x_2}, 1, \mathbf{1}[x_1 \geq 0], \mathbf{1}[x_2 \geq 0] \right\} \tag{6}$$

For each subset of features $F \subseteq \mathcal{F}$, let $D(F)$ be the set of all decision boundaries corresponding to linear classifiers that use features F.

For each of the following sets of decision boundaries E, provide the minimal F such that $D(F) \supseteq E$. If no such F exists, write 'none'.

- E is all lines:

  _____ (7)

- E is all circles centered at the origin:

  _____ (8)

- E is all circles:

  _____ (9)

- E is all axis-aligned rectangles:

  _____ (10)

- E is all axis-aligned rectangles whose lower-right corner is at $(0, 0)$:

  _____ (11)

Solution

- Lines: $x_1, x_2, 1$ $(ax_1 + bx_2 + c = 0)$
- Circles centered at the origin: $x_1^2, x_2^2, 1$ $(x_1^2 + x_2^2 = r^2)$
- Circles centered anywhere in the plane: $x_1^2, x_2^2, x_1, x_2, 1$ $((x_1 - a)^2 + (x_2 - b)^2 = r^2)$
- Axis aligned rectangles: not possible (need features of the form $\mathbf{1}[x_1 \leq a]$)
- Axis aligned rectangles with lower right corner at (0, 0): not possible

e. (*10 points*)

For each of the following problems, circle all correct options. There may be multiple correct options for a particular question.

- Hyperparameters should be tuned to minimize error on the
  - Training set
  - Development set
  - Test set
- The majority algorithm (which outputs the most common label based on the training data) for classification has
  - High bias
  - High variance
  - Low bias
  - Low variance
- Suppose our learning algorithm for a particular task has very low training error but large test error. Which of the following changes to our algorithm could fix this?
  - Increasing the number of features
  - Decreasing the number of features
  - Increasing the number of SGD iterations
  - Decreasing the number of SGD iterations
  - Adding more training examples
  - Adding a regularization penalty

Solution

- Hyperparameters are modified to minimize error on the
  - Development set
- The majority algorithm for learning suffers from
  - High bias
  - Low variance
- Suppose our learning algorithm for a particular task has very low training error but large test error. Which of the following changes to our algorithm could fix this?
  - Decreasing the number of features
  - Decreasing the number of SGD iterations
  - Adding more training examples
  - Adding a regularization penalty

2. States (*50 points*)

Sabina has just moved to a new town, which is represented as a grid of locations (see below). She needs to visit various shops $S_1, \ldots, S_k$. From a location on the grid, Sabina can move to the location that is immediately north, south, east, or west, but certain locations have been blocked off and she cannot enter them. It takes one unit of time to move between adjacent locations. Here is an example layout of the town:

	(2,5)	(3,5)	(4,5)	
(1,4)	**S_1** (2,4)	(3,4)	**S_2** (4,4)	(5,4)
(1,3)	(2,3)		(4,3)	(5,3)
	(2,2)	(3,2)	(4,2)	**S_3** (5,2)
House (1,1)	(2,1)	**S_4** (3,1)	(4,1)	(5,1)

Sabina lives at $(1, 1)$, and no location contains more than one building (Sabina's house or a shop).

a. (*10 points*)

Sabina wants to start at her house, visit the shops $S_1, \ldots, S_k$ **in any order**, and then return to her house as quickly as possible. We will construct a search problem to find the fastest route for Sabina. Each state is modeled as a tuple $s = (x, y, A)$, where (x, y) is Sabina's current position, and A is some auxiliary information that you need to choose. If an action is invalid from a given state, set its cost to infinity. Let V be the set of valid (non-blocked) locations; use this to define your search problem. You may assume that the locations of the k shops are known. You must choose a minimal representation of A and solve this problem for general k. Be precise!

- Describe A: _____
- $s_{\text{start}} =$ _____
- Actions$((x, y, A)) = \{\text{N}, \text{S}, \text{E}, \text{W}\}$

- $\text{Succ}((x, y, A), a) =$ _____

_____

- $\text{Cost}((x, y, A), a) =$ _____

- $\text{IsGoal}((x, y, A)) =$ _____

Now we will tweak the problem. Sabina must visit the shops $S_1, S_2, \ldots, S_k$ **in that order**. She is allowed to step in the location of a shop without visiting it. Assuming that each state is still (x, y, A), how would you modify A to solve the constrained version of this problem? Again, make A as minimal as possible and be precise.

Solution

- $A = (A_1, A_2, \ldots A_k)$ where $A_i \in \{0, 1\}$ is a boolean representing whether S_i has been visited or not.
- $s_{\text{start}} = (1, 1, [0, \ldots, 0])$
- $$\text{Succ}((x, y, A), a) = \begin{cases} (x, y+1, A') & \text{if } a = \text{N} \\ (x, y-1, A') & \text{if } a = \text{S} \\ (x+1, y, A') & \text{if } a = \text{E} \\ (x-1, y, A') & \text{if } a = \text{W}, \end{cases}$$

  where A' is defined as follows: $A'_i = 1$ if Sabina's new location contains shop i; otherwise, $A'_i = A_i$.
- Let $(x', y', A') = \text{Succ}((x, y, A), a)$. Then $\text{Cost}((x, y, A)) = 1$ if $(x', y') \in V$ and ∞ otherwise.
- $\text{IsGoal}((x, y, A)) = [x = 1 \wedge y = 1 \wedge A = [1, \ldots, 1]]$.

Solution If Sabina must visit the shops in order, we only need to keep track of the prefix of shops that Sabina has visited. Let $0 \leq A \leq k$ denote the fact that Sabina has visited shops $S_1, \ldots, S_A$. Note that the number of possible values of A (and hence the number of states) is much smaller now: $O(k)$ rather than $O(2^k)$. Just for completeness, we sketch the state space:

- $s_{\text{start}} = (1, 1, 0)$
- The successor function $\text{Succ}(x, y, A) = (x', y', A')$, where the prefix is updated $A' = A + 1$ if $(x', y') = S_{A+1}$ and $A' = A$ otherwise.
- $\text{IsGoal}((x, y, A)) = [x = 1 \wedge y = 1 \wedge A = k]$.

b. (*10 points*)

Sabina is now again allowed to visit the shops **in any order**. But she is impatient and doesn't want to wait around for your search algorithm to finish running. In response, you will use the A* algorithm, but you need a heuristic. For each pair of shops (S_i, S_j) where $i \neq j$ and $1 \leq i, j \leq k$, define a **consistent** heuristic $h_{i,j}$ that approximates the time it takes to ensure that shops S_i and S_j are visited and then return home. Computing $h_{i,j}(s)$ should take $O(1)$ time.

Now, let each $h_{i,j}$ be an arbitrary consistent heuristic. Let $h(s)$ be the heuristic that takes a weighted sum of these heuristics; that is, $h(s) = \sum_{i \neq j} \alpha_{i,j} h_{i,j}(s)$, for some numbers $\alpha_{i,j} \geq 0$. For which values of $\alpha_{i,j}$ is the heuristic h **guaranteed** to be consistent?

Solution We will define $h_{i,j}(x, y, A)$ now: based on A, we will have visited some subset of shops in $\{S_i, S_j\}$. We need to compute the distance to visit the remaining shops and return home. Let $d(p, q)$ refer to the Manhattan distance between points p and q, which can be computed in $O(1)$ time. Note that if we haven't visit either S_i or S_j, we must take the min over visiting either one first, in order to produce a consistent heuristic.

$$h_{i,j}(x, y, A) = \begin{cases} \min\{d((x,y), S_i) + d(S_i, S_j) + d(S_j, (1,1)), & \\ \qquad d((x,y), S_j) + d(S_j, S_i) + d(S_i, (1,1))\} & \text{if } A_i = 0 \wedge A_j = 0, \\ d((x,y), S_j) + d(S_j, (1,1)) & \text{if } A_i = 1 \wedge A_j = 0, \\ d((x,y), S_i) + d(S_i, (1,1)) & \text{if } A_i = 0 \wedge A_j = 1, \\ d((x,y), (1,1)) & \text{if } A_i = 1 \wedge A_j = 1. \end{cases}$$

It is clear that computing $h_{i,j}(x, y, A)$ is $O(1)$ it makes $O(1)$ calls to $d(\cdot, \cdot)$.

Solution Recall the definition of consistency of a heuristic h: for any s, a and $s' = \text{Succ}(s, a)$, we must have that $\text{Cost}(s, a) + h(s') - h(s) \geq 0$. Expanding h, we need that

$$\text{Cost}(s, a) + \sum_{i \neq j} \alpha_{i,j}(h_{i,j}(s') - h_{i,j}(s)) \geq 0,$$

given that we know that for each $i \neq j$ (by assumption):

$$\text{Cost}(s, a) + (h_{i,j}(s') - h_{i,j}(s)) \geq 0.$$

This holds exactly when $\sum_{i \neq j} \alpha_{i,j} \leq 1$ (remember that we already have $\alpha_{i,j} \geq 0$), so we are just taking a convex combination of a set of consistent heuristics and scaling down, which is safe to do without violating consistency.

c. (*10 points*)

Little did Sabina realize what a strange town she had moved to. It seems like whenever she tries to visit a shop, it's closed! Let's model this as a two-step game: Sabina moves first, choosing either shop 1 or shop 2. Then each shop i decides to open for Sabina with probability p_i. If Sabina visits shop i and it is open, then she gets utility $10i$. If it is closed, then she gets utility 0.

- What is the expected utility of the game? For what values of p_1 and p_2 would Sabina visit shop 1?

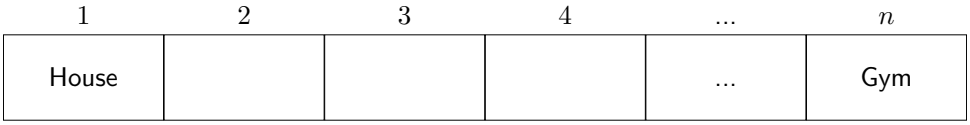
- Suppose that at the town, you get to make one of the shops adversarial and the other shop open with probability $\frac{1}{2}$. Which shop would you make adversarial to minimize the expected utility for Sabina (assuming Sabina will know which shop you choose to be the adversary)?

Solution The expected utility is $\max\{10p_1, 20p_2\}$. Sabina chooses shop 1 if $10p_1 > 20p_2$, which happens when $p_1 > 2p_2$.

Solution If we make shop 1 adversarial, then Sabina would choose shop 2 and get $\frac{1}{2}(20)$. If we make shop 2 adversarial, then Sabina would choose shop 1 and get $\frac{1}{2}(10)$. Therefore, we should make shop 2 adversarial, in which case Sabina would choose shop 1 and get expected utility 5.

d. (*10 points*)

Sabina wants to go from her house (located at 1) to the gym (located at n). At each location s, she can either (i) deterministically walk forward to the next location $s+1$ (takes 1 unit of time) or (ii) wait for the bus. The bus comes with probability ϵ, in which case, she will reach the gym in $1+\alpha(n-s)$ units of time, where α is some parameter. If the bus doesn't come, well, she stays put, and that takes 1 unit of time.



We have formalized the problem as an MDP for you:

- State: $s \in \{0, 1, \ldots, n\}$ is Sabina's location
- $\text{Actions}(s) = \{\text{Walk}, \text{Bus}\}$
- $\text{Reward}(s, \text{Walk}, s') = \begin{cases} -1 & \text{if } s' = s+1 \\ -\infty & \text{otherwise} \end{cases}$
- $\text{Reward}(s, \text{Bus}, s') = \begin{cases} -1-\alpha(n-s) & \text{if } s' = n \\ -1 & \text{if } s' = s \\ -\infty & \text{otherwise} \end{cases}$
- $T(s, \text{Walk}, s') = \begin{cases} 1 & \text{if } s' = s+1 \\ 0 & \text{otherwise} \end{cases}$
- $T(s, \text{Bus}, s') = \begin{cases} \epsilon & \text{if } s' = n \\ 1-\epsilon & \text{if } s' = s \\ 0 & \text{otherwise} \end{cases}$
- $\text{IsEnd}(s) = \mathbf{1}[s = n]$

Compute a closed form expression for the value of the "always walk" policy and the "always wait for the bus" policy (using some or all of the variables ϵ, α, n).

- $V_{\text{Walk}}(s) =$ _____

- $V_{\text{Bus}}(s) =$ _____

- For what values of ϵ (as a function of α and n) is it advantageous to walk rather than take the bus?

Solution Expected value for always walking:

$$V_{\text{Walk}} = -(n-s).$$

Expected value for always waiting for bus:

$$V_{\text{Bus}}(s) = \epsilon(-1-\alpha(n-s)) + (1-\epsilon)(-1+V_{\text{Bus}}(s)).$$

Simplifying, we get:

$$V_{\text{Bus}}(s) = -\alpha(n-s) - \frac{1}{\epsilon}.$$

For walking to be preferable, we need $V_{\text{Walk}}(s) \geq V_{\text{Bus}}(s)$, or equivalently:

$$n-s < \alpha(n-s) + \frac{1}{\epsilon} \Leftrightarrow (1-\alpha)(n-s) < \frac{1}{\epsilon} \Leftrightarrow \begin{cases} \epsilon < \frac{1}{(1-\alpha)(n-s)} & ,\alpha < 1 \\ \epsilon > 0 & ,\alpha \geq 1. \end{cases}$$

e. (*10 points*)

Not surprisingly, buses operate strangely in this town, and we will now assume instead that Sabina doesn't know the reward function nor the transition probabilities. She decides to use reinforcement learning to find out. She starts by going around town using the two different modes of transportation:

s_0	a_1	r_1	s_1	a_2	r_2	s_2	a_3	r_3	s_3	a_4	r_4	s_4	a_5	r_5	s_5
1	Bus	-1	1	Bus	-1	1	Bus	3	3	Walk	1	4	Walk	1	5

Run the Q-learning algorithm once over the given data to compute an estimate of the optimal Q-value $Q_{\text{opt}}(s, a)$. Process the episodes from left to right. Assume all Q-values are initialized to zero, and use a learning rate of $\eta = 0.5$ and a discount of $\gamma = 1$.

- $\hat{Q}(1, \text{Walk}) =$ _____
- $\hat{Q}(1, \text{Bus}) =$ _____
- $\hat{Q}(3, \text{Walk}) =$ _____
- $\hat{Q}(3, \text{Bus}) =$ _____
- $\hat{Q}(4, \text{Walk}) =$ _____
- $\hat{Q}(4, \text{Bus}) =$ _____

Solution On each (s, a, r, s'), recall the Q-learning updates:

$$\hat{Q}_{opt}(s, a) \leftarrow (1 - \eta)\hat{Q}_{opt}(s, a) + \eta(r + \gamma \max_{a' \in \text{Actions}(s')} \hat{Q}_{opt}(s', a')). \tag{12}$$

Now the concrete updates:

- On $(1, \text{Bus}, -1, 1)$: $\hat{Q}(1, \text{Bus}) = 0.5(0) + 0.5(-1 + 1(\max(0, 0))) = -0.5$
- On $(1, \text{Bus}, -1, 1)$: $\hat{Q}(1, \text{Bus}) = 0.5(-0.5) + 0.5(-1 + 1(\max(0, -0.5))) = -0.75$
- On $(1, \text{Bus}, 3, 3)$: $\hat{Q}(1, \text{Bus}) = 0.5(-0.75) + 0.5(3 + 1(\max(0, 0))) = 1.125$
- On $(3, \text{Walk}, 1, 4)$: $\hat{Q}(3, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0, 0))) = 0.5$
- On $(4, \text{Walk}, 1, 5)$: $\hat{Q}(4, \text{Walk}) = 0.5(0) + 0.5(1 + 1(\max(0, 0))) = 0.5$

The final values:

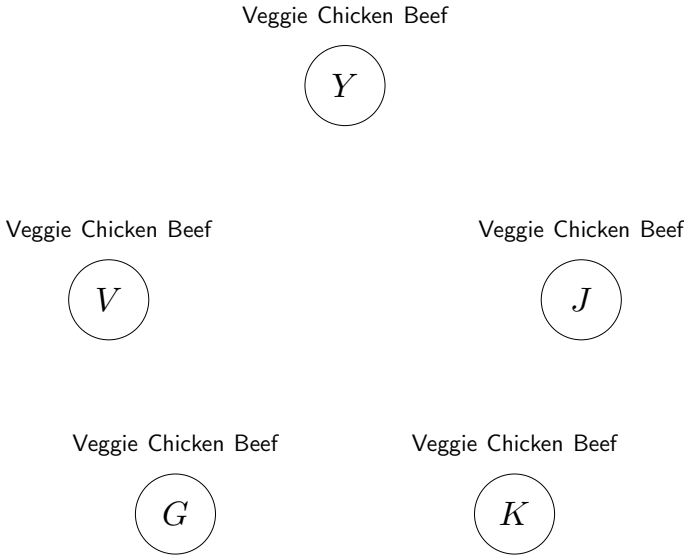
- $\hat{Q}(1, \text{Walk}) = 0$
- $\hat{Q}(1, \text{Bus}) = 1.125$
- $\hat{Q}(3, \text{Walk}) = 0.5$
- $\hat{Q}(3, \text{Bus}) = 0$
- $\hat{Q}(4, \text{Walk}) = 0.5$
- $\hat{Q}(4, \text{Bus}) = 0$

3. Variables and Logic (*50 points*)

It's Friday night, and you and your friends go out to dinner in anticipation of the Big Game the next day (go Card!). Since you've just been working on your final project for CS221, you are seeing CSPs and Bayesian networks everywhere!

a. (*10 points*)

You and your friends (Veronica, Jarvis, Gabriela, Kanti) sit around a table like this:



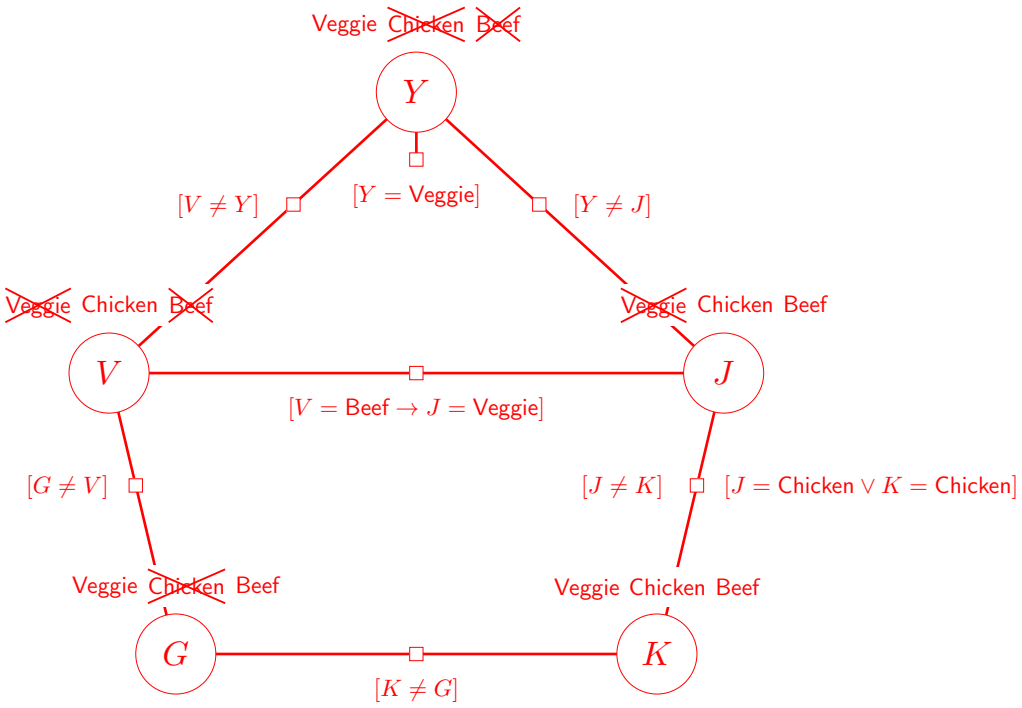
There are three dishes on the menu: the vegetarian deep dish pizza, the chicken quesadilla, and the beef cheeseburger. Each person will order exactly one dish.

But what started out as a simple dinner has quickly turned into a logistical nightmare because of all the constraints you and your friends impose upon yourselves:

1. Each person must order something different than the people sitting immediately next to him/her.
2. You (Y) are vegetarian.
3. If Veronica (V) orders beef, then Jarvis (J) will order veggie.
4. Kanti (K) and Jarvis (J) cannot both get non-chicken dishes.

Draw the potentials for the above constraints and write the propositional formula above each potential (e.g., $[Y = \text{Veggie}]$). Then for each pair of variables, enforce arc consistency in both directions, crossing out the appropriate values from the domains.

Solution

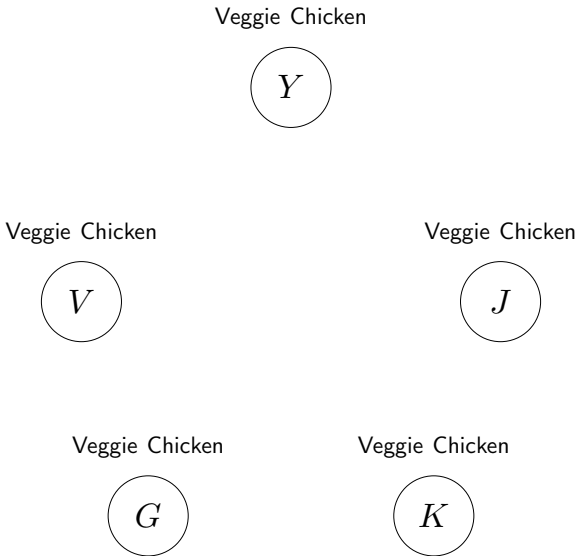

Veggie Chicken Beef
Y
[Y = Veggie]
[V ≠ Y]
[Y ≠ J]
Veggie Chicken Beef
Veggie Chicken Beef
V
J
[V = Beef → J = Veggie]
[G ≠ V]
[J ≠ K]
[J = Chicken ∨ K = Chicken]
Veggie Chicken Beef
Veggie Chicken Beef
G
K
[K ≠ G]

b. (*10 points*)

Your server comes by your table and tells you that they are out of beef today, so you all decide to rework your constraints. Now they are:

1. There is a preference for people sitting next to each other to order different dishes. Formally, we have 5 potentials: $f(Y, J) = \mathbf{1}[Y \neq J] + 1$, $f(J, K) = \mathbf{1}[J \neq K] + 1$, etc.
2. You (Y) are vegetarian.

With the 2 constraints above, what is a maximum weight assignment and what is its weight? If there are many assignments with the same maximum weight, give any one. For convenience, the updated table is given below.



Solution The maximum weight 16. Many answers can be correct. They just need to satisfy the following: 1. Y=Veggie; 2. Only one pair of adjacent people have the same dish. One max weight assignment is:

Y	Veggie
V	Chicken
G	Veggie
K	Chicken
J	Chicken

c. (*10 points*)

Now that you and your friends have your food, you are watching the highlights from yesterday's football game. As you watch a particular four-down sequence, you have an epiphany: football is a glorified Bayesian network.

On each play $t \in \{1, 2, 3, \dots\}$, the quarterback will either (1) run the ball, (2) throw a short pass, or (3) throw a long pass. Let $X_t \in \{1, 2, 3\}$ denote the quarterback's move, and E_t be the number of yards advanced as a result. Assume that X_1 is chosen uniformly at random from $\{1, 2, 3\}$.

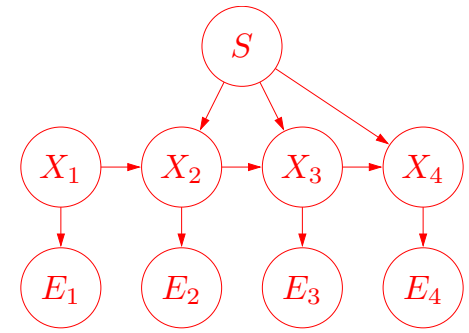
Before the game starts, the team decides on one of the two strategies ($S \in \{A, B\}$), which is chosen uniformly at random and is held constant throughout the game: if $S = A$, then the quarterback uses $\alpha = \frac{1}{2}$; if $S = B$, then the quarterback uses $\alpha = \frac{1}{4}$.

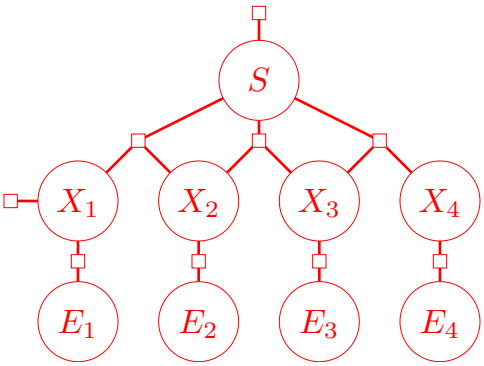
On each t, the quarterback performs the same move as the last time ($X_t = X_{t-1}$) with probability α, and one of the other two each with probability $\frac{1-\alpha}{2}$. When a play $x \in \{1, 2, 3\}$ is made, the ball is advanced $5x$ yards with probability 2^{-x} and 0 yards with probability $1 - 2^{-x}$.

- Draw the Bayesian network for corresponding to the football model described above, using $t \in \{1, 2, 3, 4\}$. Recall that the variables are $S, X_1, \dots, X_4, E_1, \dots, E_4$.

- Draw the factor graph associated with the Bayesian network corresponding to this football model. You are not required to write the potentials for each factor.

Solution





d. (*10 points*)

You saw that the first play was a long pass ($X_1 = 3$) which resulted in advancing the ball $E_1 = 15$ yards (this happened with probability $\frac{1}{8}$). Then while you were busy constructing your Bayesian network, you missed what happened in the second time step, but you found out that only five yards were gained on the third play ($E_3 = 5$). You do not remember the yard line the ball was previously at, so you have no knowledge about what type of play was run on the second down or what its result was. What is the probability that the team is playing strategy A? In other words, compute:

$$\mathbb{P}(S = A \mid X_1 = 3, E_1 = 15, E_3 = 5).$$

Solution First, we can remove (marginalize out) E_1, E_2, X_4, E_4 since they are downstream of all the variables mentioned in the queries. Second, conditioned on $E_3 = 5$, we know that $X_3 = 1$ (note that this doesn't happen generally in Bayesian networks, but for these particular local conditional distributions, things worked out nicely). This renders all the previous time steps independent. So we just need to marginalize out X_2:

$$P(S = s \mid X_1 = 3, E_1 = 15, E_3 = 5) \propto p_S(s) \sum_{x_2} p(x_2 \mid x_1 = 3, s) p(x_3 = 1 \mid x_2, s).$$

Since $P(S = A) = P(S = B)$, we have

$$P(S = A \mid X_1 = 3, E_1 = 15, E_3 = 5) = \frac{P(X_3 = 1 | X_1 = 3, S = A)}{P(X_3 = 1 | X_1 = 3, S = A) + P(X_3 = 1 | X_1 = 3, S = B)}$$

For $\lambda \in \{A, B\}$, using $\alpha_A = \frac{1}{2}, \alpha_B = \frac{1}{4}$

$$\begin{aligned} P(X_3 = 1 | X_1 = 3, S = \lambda) &= \sum_{i=1}^{3} P(X_3 = 1 | X_2 = i, S = \lambda) \cdot P(X_2 = i | X_1 = 3, S + \lambda) \\ &= 2\alpha_\lambda \frac{1 - \alpha_\lambda}{2} + \left(\frac{1 - \alpha_\lambda}{2}\right)^2 \\ &= -\frac{3}{4}{\alpha_\lambda}^2 + \frac{1}{2}\alpha_\lambda + \frac{1}{4} \end{aligned}$$

Thus $P(X_3 = 1 | X_1 = 3, S = A) = \frac{5}{16}$, $P(X_3 = 1 | X_1 = 3, S = B) = \frac{21}{64}$
Thus $P(S = A \mid X_1 = 3, E_1 = 15, E_3 = 5) = \frac{5}{16} / (\frac{5}{16} + \frac{21}{64}) = \frac{20}{41}$

e. (*10 points*)

You realize that the football model you made up was pretty lousy, and it's better to learn the model from data. You will use the same Bayesian network as before, but now just learn the weights. In this new model, any play can gain either 0, 5, or 10 yards. Teams still play with one of the two strategies: strategy $S = A$ and strategy $S = B$, although they may not have the same interpretation they had before. As more highlights flash across the screen, the sportscaster announces the strategy and you write the plays down furiously on your napkin:

- Strategy A: $(2, 0), (1, 0)$
- Strategy B: $(1, 5), (1, 0), (1, 10), (3, 0)$
- Strategy A: $(3, 5), (2, 0), (1, 10), (1, 0), (3, 10)$

This means that the first sequence was with strategy A, which involved two plays: (short pass, 0 yardage) and (run, 0 yardage). The next sequence involved strategy B, etc. Each sequence is independent.

Use Laplace smoothing with $\lambda = 1$.

x_t	e_t	$p(e_t \mid x_t)$
1	0	
1	5	
1	10	
2	0	
2	5	
2	10	
3	0	
3	5	
3	10	

s	x_{t-1}	x_t	$p(x_t \mid x_{t-1}, s)$
A	1	1	
A	1	2	
A	1	3	
A	2	1	
A	2	2	
A	2	3	
A	3	1	
A	3	2	
A	3	3	
B	1	1	
B	1	2	
B	1	3	
B	2	1	
B	2	2	
B	2	3	
B	3	1	
B	3	2	
B	3	3	

Solution

x_t	e_t	$p(e_t \mid x_t)$
1	0	4/9
1	5	2/9
1	10	1/3
2	0	3/5
2	5	1/5
2	10	1/5
3	0	1/3
3	5	1/3
3	10	1/3

s	x_{t-1}	x_t	$p(x_t \mid x_{t-1}, s)$
A	1	1	2/5
A	1	2	1/5
A	1	3	2/5
A	2	1	3/5
A	2	2	1/5
A	2	3	1/5
A	3	1	1/4
A	3	2	1/2
A	3	3	1/4
B	1	1	1/2
B	1	2	1/6
B	1	3	1/3
B	2	1	1/3
B	2	2	1/3
B	2	3	1/3
B	3	1	1/3
B	3	2	1/3
B	3	3	1/3