CS221 Exam Solutions

CS221
November 17, 2015

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. Two views (*50 points*)

Alice and Bob are trying to predict movie ratings. They cast the problem as a standard regression problem,[1] where $x \in \mathcal{X}$ contains information about the movie and $y \in \mathbb{R}$ is the real-valued movie rating.

To split up the effort, Alice will create a feature extractor $A : \mathcal{X} \to \mathbb{R}^d$ using information from the text of the movie reviews and Bob will create a feature extractor $B : \mathcal{X} \to \mathbb{R}^d$ from the movie metadata (genre, cast, number of reviews, etc.) These features will be used to do linear regression as follows: If $\mathbf{u} \in \mathbb{R}^d$ is a weight vector associated with A and $\mathbf{v} \in \mathbb{R}^d$ is a weight vector associated with B, then we can define the resulting predictor as:

$$f_{\mathbf{u},\mathbf{v}}(x) \stackrel{\text{def}}{=} \mathbf{u} \cdot A(x) + \mathbf{v} \cdot B(x). \tag{1}$$

We are interested in the squared loss:

$$\text{Loss}(x, y, \mathbf{u}, \mathbf{v}) \stackrel{\text{def}}{=} (f_{\mathbf{u},\mathbf{v}}(x) - y)^2. \tag{2}$$

Let $\mathcal{D}_{\text{train}}$ be the training set of (x, y) pairs, on which we can define the training loss:

$$\text{TrainLoss}(\mathbf{u}, \mathbf{v}) \stackrel{\text{def}}{=} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{u}, \mathbf{v}). \tag{3}$$

[1] Remember that you should be going beyond this for your project!

a. (*10 points*) Compute the gradient of the training loss with respect to Alice's weight vector $\mathbf{u}$.

$\nabla_{\mathbf{u}} \text{TrainLoss}(\mathbf{u}, \mathbf{v}) =$ _____

Solution

$$\nabla_{\mathbf{u}} \text{TrainLoss}(\mathbf{u}, \mathbf{v}) = 2 \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (f_{\mathbf{u},\mathbf{v}} - y) A(x) \tag{4}$$

$$= 2 \sum_{(x,y) \in \mathcal{D}_{\text{train}}} (\mathbf{u} \cdot A(x) + \mathbf{v} \cdot B(x) - y) A(x). \tag{5}$$

Suppose Alice and Bob have found a current pair of weight vectors $(\mathbf{u}, \mathbf{v})$ where the gradient with respect to $\mathbf{u}$ is zero: $\nabla_{\mathbf{u}} \text{TrainLoss}(\mathbf{u}, \mathbf{v}) = 0$. Mark each of the following statements as true or false (no justification is required).

1. Even if Bob updates his weight vector $\mathbf{v}$ to $\mathbf{v}'$, $\mathbf{u}$ is still optimal for the new $\mathbf{v}'$; that is, $\nabla_{\mathbf{u}} \text{TrainLoss}(\mathbf{u}, \mathbf{v}') = 0$ for any $\mathbf{v}'$.
   – True / False

2. The gradient of the loss on each example is also zero: $\nabla_{\mathbf{u}} \text{Loss}(x, y, \mathbf{u}, \mathbf{v}) = 0$ for each $x, y \in \mathcal{D}_{\text{train}}$.
   – True / False

Solution Neither statement is true:

1. Alice's weight vector $\mathbf{u}$ is only optimal with respect to the particular $\mathbf{v}$. As another example, think of alternating minimization (k-means), where the centroids are only optimal *given* the current assignments, not for all assignments; otherwise, there would be no point in iterating.

2. The loss gradient is only zero *on average*, definitely not for each example.

b. (*10 points*)

Alice and Bob got into an argument and now aren't on speaking terms. Each of them therefore trains his/her weight vector separately: Alice computes

$$\mathbf{u}_A = \arg\min_{\mathbf{u}\in\mathbb{R}^d} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} (\mathbf{u}\cdot A(x) - y)^2$$

and Bob computes

$$\mathbf{v}_B = \arg\min_{\mathbf{v}\in\mathbb{R}^d} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} (\mathbf{v}\cdot B(x) - y)^2.$$

Just for reference, if they had worked together, then they would have gotten:

$$(\mathbf{u}_C, \mathbf{v}_C) = \arg\min_{\mathbf{u},\mathbf{v}} \text{TrainLoss}(\mathbf{u}, \mathbf{v}).$$

Let L_A, L_B, L_C be the training losses of the following predictors:

- (L_A) Alice works alone: $x \mapsto \mathbf{u}_A \cdot A(x)$
- (L_B) Bob works alone: $x \mapsto \mathbf{v}_B \cdot B(x)$
- (L_C) They work together from the beginning: $x \mapsto \mathbf{u}_C \cdot A(x) + \mathbf{v}_C \cdot B(x)$

For each statement, mark it as necessarily true, necessarily false, or neither:

1. $L_A \le L_B$ — True / False / Neither
2. $L_C \le L_A$ — True / False / Neither
3. $L_C \le L_B$ — True / False / Neither

Solution We can view L_A as the resulting training loss from optimizing over all weight vectors $(\mathbf{u}, 0)$, whereas L_C optimizes over all $(\mathbf{u}, \mathbf{v})$; therefore $L_C \le L_A$. The same logic applies to conclude $L_C \le L_B$. However, we cannot make a comparison between L_A and L_B; it's possible that either Alice or Bob's features get lower training error.

Now, let L'_A, L'_B, L'_C be the corresponding losses on the *test set*. For each statement, mark it as necessarily true, necessarily false, or neither:

1. $L'_A \le L'_B$ — True / False / Neither
2. $L'_C \le L'_A$ — True / False / Neither
3. $L'_C \le L'_B$ — True / False / Neither

Solution Neither for all three! We might hope that $L'_C \le L'_A$ and $L'_C \le L'_A$, but we might have overfit on the training set, and the same trends might not hold.

c. (*10 points*)
Alice and Bob have reconciled, and together they collected a small training set:[2]

	Review text	# reviews	Genre	Rating
Example 1	great movie	100	action	5
Example 2	great quality	10	action	4
Example 3	poor quality	10	action	2
Example 4	poor movie	1	action	1

Alice and Bob wrote down the following features:

- (Alice) review text contains "great"
- (Alice) review text contains "poor"
- (Alice) review text contains "quality"
- (Alice) review text contains "movie"
- (Bob) # reviews
- (Bob) $\log_{10}$(# reviews)
- (Bob) genre is "comedy"
- (Bob) 1 (bias feature)

For example, if you set all feature weights to 1, then the prediction on example 1 is $1 + 0 + 0 + 1 + 100 + \log_{10}(100) + 0 + 1 = 105$, and the squared loss on that example would be $(105 - 5)^2 = 10000$ (not so good!). For the following three questions, try to use your intuition about linear models; don't brute force all possibilities.

[2]For your final project, you hopefully have more data than they do!

(i) Alice decides to choose exactly *one* of her features in a linear model. Which feature should she choose and what is the associated feature weight in order to obtain the smallest total squared loss across all training examples? You don't need to give the value of the squared loss.

Alice's Feature: _____

Feature weight: _____

Solution Intuitively, we want to choose features to model the movies with ratings far from zero to reduce the loss. Set the weight of "great" to 4.5; then the predictions by the model are 4.5, 4.5, 0, 0. This yields an squared loss of $2 \cdot 0.5^2 + 2^2 + 1^1 = 5.5$.

(ii) Similarly, Bob decides to choose exactly *one* of his features. Which feature should he choose and what is the associated feature weight in order to obtain the smallest total squared loss across all training examples? You don't need to give the value of the squared loss.

Bob's Feature: _____

Feature weight: _____

Solution Let w be the weight of $\log_{10}(\#\text{ reviews})$. The prediction error is $(2w-5)^2 + (w-4)^2 + (w-2)^2 + (0-1)^2$. Differentiating and setting to zero, we get the condition $(2w-5)\cdot 2 + (w-4) + (w-2) = 0$, which simplifies to $6w - 16 = 0$, which has the solution $w = 8/3$.

(iii) What is the smallest set of features from both Alice and Bob's set that would allow us to get 0 prediction error? What are the weights of the chosen features?

Feature set: _____

Feature weights: _____

Solution We just need the following features:

- review text contains "great": 2
- $\log_{10}$(# reviews): 1
- 1 (bias feature): 1

d. (*10 points*)

Alice and Bob were so excited about their dataset and feature extractors, that they immediately started running feature extraction. For each original training example (x_i, y_i) $(i = 1, \ldots, n)$, they computed Alice's feature vector $A(x_i)$ and Bob's feature vector $B(x_i)$. But they had a bug in their code that caused them to switch the feature vectors once in a while. Specifically, their code overwrote the original example (x_i, y_i) with $(A(x_i), B(x_i), y_i)$ some of the time and $(B(x_i), A(x_i), y_i)$ some of the time. Whoops![3]

They think for a bit and develop an ingenious scheme to take into account this shuffling. The intuition is that they will try treating each example (c_i, d_i, y_i) as either $(A(x_i), B(x_i), y_i)$ or $(B(x_i), A(x_i), y_i)$ depending on which one yields lower loss. Here's the objective function they wrote down:

$$\text{ShuffledTrainLoss}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} \min\{(\mathbf{u} \cdot c_i + \mathbf{v} \cdot d_i - y_i)^2, (\mathbf{u} \cdot d_i + \mathbf{v} \cdot c_i - y_i)^2\}. \quad (6)$$

Help Alice and Bob derive an alternating minimization algorithm. Let $z_i \in \{0, 1\}$ denote whether the feature vectors are swapped or not for each $i = 1, \ldots, n$. Specifically, $z_i = 0$ corresponds to the case where $c_i = A(x_i)$ and $d_i = B(x_i)$; $z_i = 1$ corresponds to the case where $d_i = A(x_i)$ and $c_i = B(x_i)$.

(i) Write the closed form update for each z_i given $\mathbf{u}, \mathbf{v}$ (don't worry about ties):

Solution For each z_i, we simply choose whichever permutation yields the lower squared loss:

$$z_i = [(\mathbf{u} \cdot d_i + \mathbf{v} \cdot c_i - y_i)^2 < (\mathbf{u} \cdot c_i + \mathbf{v} \cdot d_i - y_i)^2].$$

(ii) Write the optimization problem for $(\mathbf{u}, \mathbf{v})$ given $z_1, \ldots, z_n$:

Solution Given the z_i's, we have a vanilla regression problem, where the data points are (c_i, d_i, y_i) if $z_i = 0$ and (d_i, c_i, y_i) if $z_i = 1$. The optimization problem is finding the $\mathbf{u}$ and $\mathbf{v}$ that minimize the following objective function:

$$\text{ShuffledTrainLoss}(\mathbf{u}, \mathbf{v}) = \sum_{i:z_i=0} (\mathbf{u} \cdot c_i + \mathbf{v} \cdot d_i - y_i)^2 + \sum_{i:z_i=1} (\mathbf{u} \cdot d_i + \mathbf{v} \cdot c_i - y_i)^2). \quad (7)$$

$$(\mathbf{u}, \mathbf{v}) = \arg\min_{\mathbf{u},\mathbf{v}} \text{ShuffledTrainLoss}(\mathbf{u}, \mathbf{v}) \quad (8)$$

---

[3]For your final project, always save and backup your raw data.

e. (*10 points*)

Alice and Bob realize that it wasn't a bug at all, but it was Competitive Carol from another group up to no good! Here's what Carol was doing: for each example $i = 1, \ldots, n$, she would adversarily choose to swap the features or not so that Alice and Bob would get the worst (highest) loss. Worse, Carol had found out a way to do the same adversarial swapping at test time too.

After suffering ten minutes of standing with their mouths agape in horror, Alice and Bob eventually wrote down the following objective function to model Carol's machinations:

$$\text{AdversarialTrainLoss}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} \max\{(\mathbf{u} \cdot c_i + \mathbf{v} \cdot d_i - y)^2, (\mathbf{u} \cdot d_i + \mathbf{v} \cdot c_i - y)^2\}. \tag{9}$$

Compute the gradient $\nabla_{\mathbf{u}}\text{AdversarialTrainLoss}(\mathbf{u}, \mathbf{v})$ with respect to Alice's weight vector $\mathbf{u}$ (Bob's would be similar, so you don't need to do it). For your convenience, to keep your equations compact, define:

$$r_i = \mathbf{u} \cdot c_i + \mathbf{v} \cdot d_i - y \tag{10}$$

$$s_i = \mathbf{u} \cdot d_i + \mathbf{v} \cdot c_i - y. \tag{11}$$

$\nabla_{\mathbf{u}}\text{AdversarialTrainLoss}(\mathbf{u}, \mathbf{v}) =$

Solution

$$\nabla_{\mathbf{u}}\text{AdversarialTrainLoss}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{n} \nabla_{\mathbf{u}}\text{AdversarialLoss}_i(\mathbf{u}, \mathbf{v}), \tag{12}$$

where

$$\nabla_{\mathbf{u}}\text{AdversarialLoss}_i(\mathbf{u}, \mathbf{v}) = \begin{cases} 2r_i c_i & \text{if } r_i^2 \geq s_i^2 \\ 2s_i d_i & \text{otherwise.} \end{cases} \tag{13}$$

2. Rafting (*50 points*)

You are going on a rafting trip! The river is modeled as a grid of positions (x, y), where $x \in \{-m, -(m-1), \ldots, 0, \ldots, (m-1), m\}$ represents the horizontal offset from the middle of the river and $y \in \{0, \ldots, n\}$ is how far down the river you are. To make things more challenging, there are a number of rocks in the river: For each position (x, y), let $R(x, y) = 1$ if there is a rock and 0 otherwise. You can assume that the start and end positions do not have rocks.

Here's how you can control the raft. From any position (x, y), you can:

- go straight down to $(x, y+1)$ (which takes 1 second),
- veer left to $(x-1, y+1)$ (which takes 2 seconds), or
- veer right to $(x+1, y+1)$ (which takes 2 seconds).

If the raft enters a position with a rock, there is an extra 5 second delay. The raft starts in $(0, 0)$ and you want to end up in any position (x, y) where $y = n$.

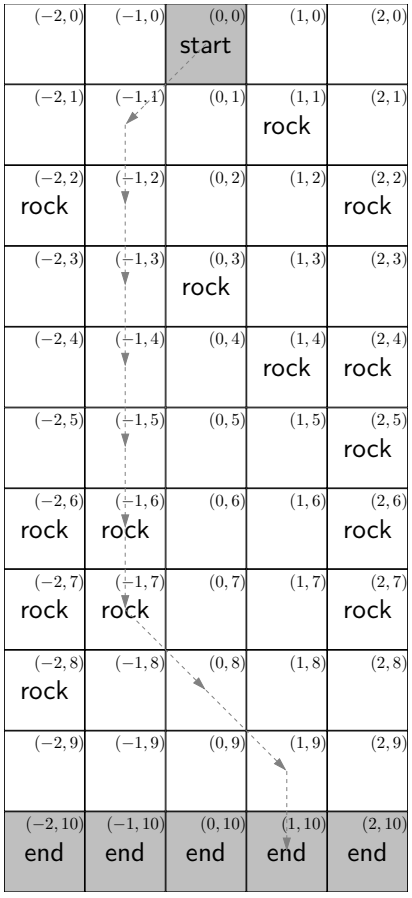


Figure 1: The goal is to raft down the river from 'start' to one of the 'end' positions in the least amount of time. For this example, $m = 2, n = 10$.

a. (*10 points*)

Let $C(x, y)$ denote the minimum time that it takes to get from position (x, y) to an end position. For convenience, let $C(x, y) = \infty$ for invalid positions (x, y) (those that are off the grid).

Write down the dynamic programming recurrence (you should have a base case and a recursive case):

Solution At each point in time, you can take an action $a \in \{-1, 0, +1\}$, which has cost $|a|+1$ (just a succinct way of representing the cost). In addition, you incur an extra $5R(x, y)$ for hitting a rock. The full recurrence is:

$$C(x, y) = 5R(x, y) + \begin{cases} 0 & \text{if } y = n, \\ \min\limits_{a \in \{-1,0,+1\}} (|a| + 1 + C((x + a, y + 1))) & \text{otherwise.} \end{cases}$$

$C(x, y) =$ _____

b. (*10 points*)

During the rafting trip, you are planning to raft down not just one $m \times n$ section of the river, but K of them. You are given K maps $R_1, \ldots, R_K$, where each $R_j(x, y) = 1$ if there is a rock in position (x, y) in the j-th map. Assume that each map has at most p rocks (where p is much smalller than mn). You could find the minimum cost path for each of the K maps separately, but that's a lot of work. Upon inspection, you find that the maps are quite similar, so you suspect there's a way to save some work.

Indeed there is! Precisely define a *single* heuristic $h(x, y)$ that can be used to do A* search on each of the K maps. Your heuristic must be

- consistent in the search problem defined by *each* of the K maps,
- non-trivial, which means you must use the information from the rocks somehow (not just define something like the Manhattan distance), and
- computable in $O(kp + mn)$ time.

Write one sentence justifying why your heuristic is consistent, and write one sentence explaining the running time for computing the heuristic for each position (x, y).

Solution Remember to think of developing heuristics via relaxation: let's remove rocks! Take the intersection of all the maps $R(x, y) = \prod_{j=1}^{K} R_j(x, y)$, that is, keeping a rock only if it occurs in all K maps. Define $h(x, y)$ to be the minimum time from (x, y) to an end state under R. This heuristic can be computed by the dynamic program in part (a) in time $O(mn)$. This heuristic is consistent for all the maps because the intersection map $R(x, y)$ is a relaxed problem with respect to each of the K maps (fewer rocks means that the action costs can only go down).

c. (*10 points*)

You find out that the maps are actually all wrong (and you can't get your money back either!), so you're going to have to start rafting without full knowledge about the location of the rocks. Fortunately, you have eyes, so at position (x, y) you can see whether there's a rock at $(x', y+1)$ for all x'. Furthermore, you assume that each position has an independent probability α of having a rock. To simplify, assume that there are no rocks in the $y = 0$ and $y = 1$ rows. You want to minimize the *expected* time to reach an end goal.

Show that you can solve this problem by defining an MDP whose maximum expected utility policy minimizes the expected the time to reach an end goal. Reduce the number of states as much as you can while still maintaining optimality; for full credit, you should be able to store only $O(mn)$ states. Define precisely what your states, transition probabilities are, etc., but don't worry too much about the corner cases. (Hint: think carefully about what you actually need to remember in the state, and explicitly define the state tuple $s =$ _____.)

- $s_{\text{start}} =$
- $\text{Actions}(s) = \{a \in \{-1, 0, +1\} : \text{ moving horizontally in direction } a \text{ keeps you on the grid}\}$
- $T(s, a, s') =$
- $\text{Reward}(s, a, s') =$
- $\text{IsEnd}(s) =$
- $\gamma =$

Solution First, let us figure out what the state should be. At position (x, y), you have knowledge about each $R(x', y')$ for all x', y' for which $y' \leq y + 1$ and for all x'. This is a lot of information to remember (exponential in mn). The key is that the only relevant bits of information are $R(x', y')$ for the positions that you might move to, of which there are 3. Let $s = (x, y, r_{-1}, r_0, r_1)$, where r_a denotes whether position $(x + a, y + 1)$ has a rock or not. Thus, there are only $2^3 mn$ possible states that you can be in at any given time.

- $s_{\text{start}} = (0, 0, 0, 0, 0)$
- $\text{Actions}(s) = \{a \in \{-1, 0, +1\} : a \text{ keeps you on the grid}\}$
- $T(s, a, s') = \alpha^k (1 - \alpha)^{3-k}$ if $x' = x + a$, $y' = y + 1$ and 0 otherwise, where we define $k = r'_{-1} + r'_0 + r'_1$ to be the number of rocks that will appear next.
- $\text{Reward}(s, a, s') = -[(|a| + 1) + 5r_a]$, where the first term is the cost of taking the action and the second term is the cost (0 or 1) of hitting a rock.
- $\text{IsEnd}(s) = [y = n]$
- $\gamma = 1$

Notice that there are some spurious states (r_1 doesn't make sense when you're already at the right side of the river with $x = m$), but it doesn't affect correctness or increase the asymptotic complexity.

d. (*10 points*)

As before, assume that you don't have the map describing the position of the rocks, but you know that the probability distribution over the map is $R(x, y) = 1$ independently with probability α.

Consider the following two scenarios:

- A genie reveals the entire map to you right as you get on your raft and at that instant, you run your blazing fast code to find the minimum time path (using part (a)). Let T_1 be this expected minimum time of getting to an end goal.
- There is sadly no genie, and you have to use your own eyes to look at the position of the next row as you're rafting. But you solve the MDP from part (c) to find the best policy. Let T_2 be this minimum expected time of getting to an end goal.

Prove that $T_1 \leq T_2$. (Intuitively this should be true; you must argue it mathematically.)

Solution First, note that $\sum_a p(a) \min_b F(a, b) \leq \min_b \sum_a p(a) F(a, b)$, because in the first case you get a different b for each a and in the second, you don't. Let $R_1, \ldots, R_n$ be the rows of R, and let $a_1, \ldots, a_n$ be the n actions that you take to get from row 0 to row n. Let $C(R_1, \ldots, R_n, a_1, \ldots, a_n)$ be the cost of your journey, and let $p(R_y)$ be the probability of having a particular configuration of rocks in row y. In the first case, we are computing an expectation over a minimum:

$$\sum_{R_1,\ldots,R_n} p(R_1) \cdots p(R_n) \min_{a_1,\ldots,a_n} C(R_1, \ldots, R_n, a_1, \ldots, a_n).$$

In the second case, we are interleaving the minimum with the expectation:

$$\sum_{R_1} p(R_1) \min_{a_1} \sum_{R_2} p(R_2) \min_{a_2} \cdots \sum_{R_n} p(R_n) \min_{a_n} C(R_1, \ldots, R_n, a_1, \ldots, a_n).$$

Having all the mins on the inside can only make the expected time smaller. In other words, going second in a game is always preferable.

e. (*10 points*)

Suppose we don't actually know how long it takes to go over rocks or what the distribution of rocks is, so we will use Q-learning to learn a good policy.

- Suppose the state s includes the position (x, y) and the map that's revealed so far, i.e. $R(x', y')$ for all x' and $y' \leq y + 1$.
- The actions are $a \in \{-1, 0, +1\}$, corresponding to going left, straight, or right.
- The reward is the negative time it takes to travel from state s to the new state s'.
- Assume the discount $\gamma = 1$.

For each state s and action a, let $H(s, a) = 1$ if a causes the raft to hit a rock and 0 otherwise. Now define the approximate Q-function to be:

$$Q(s, a; \alpha, \beta) = \alpha H(s, a) + \beta,$$

where α and β are parameters to be learned. Suppose we sample once from an exploration policy, which led to the trajectory shown in Figure **??**.

(i) Write down the Q-learning updates on α on experience (s, a, r, s') using a step size η. Your formula should be in terms of $H(s, a), \alpha, \beta$ and should not contain generic RL notation.

$\alpha \leftarrow \alpha - \eta$_____

$\beta \leftarrow \beta - \eta$_____

Solution For updating α:

$$\alpha \leftarrow \alpha - \eta[(\alpha H(s, a) + \beta) - (r + \gamma \max_{a'}(\alpha H(s', a') + \beta))]H(s, a) \tag{14}$$

$$= \alpha - \eta[\alpha H(s, a) - (r + \max_{a'} \alpha H(s', a')]H(s, a). \tag{15}$$

For updating β:

$$\beta \leftarrow \beta - \eta[(\alpha H(s, a) + \beta) - (r + \gamma \max_{a'}(\alpha H(s', a') + \beta))] \tag{16}$$

$$= \beta - \eta[\alpha H(s, a) - (r + \max_{a'} \alpha H(s', a')]. \tag{17}$$

(ii) On how many of the $n = 10$ updates could α change its value?

Solution The paramater α is updated only when $H(s, a) = 1$, which means that we hit a rock. This happens twice. _____

3. Voting (*50 points*)

You decide to visit Bayesland for Thanksgiving. It happens to be prime voting season there. Having just learned about Bayesian networks, you see that the voting process (on any given issue) can be modeled by following Bayesian network (see Figure **??**):

1. The head of state makes a statement either in favor of the issue ($H = 1$) with probability $\frac{1}{2}$ or against the issue ($H = 0$) with probability $\frac{1}{2}$.

2. Each of the n citizens casts an independent vote $C_i \in \{0, 1\}$ given the head's statement, deviating from H with probability α; that is: $p(c_i \mid h) = \alpha$ if $c_i \neq h$ and $1 - \alpha$ if $c_i = h$.

3. Because of the backwards technology in Bayesland, each vote C_i gets flipped with probability β, and this noisy version D_i is sent to the central agency.

4. Finally, the central agency reports $R = [\sum_{i=1}^{n} D_i > n/2]$ as the final result, which is whether over half the of votes received were in favor of the issue.

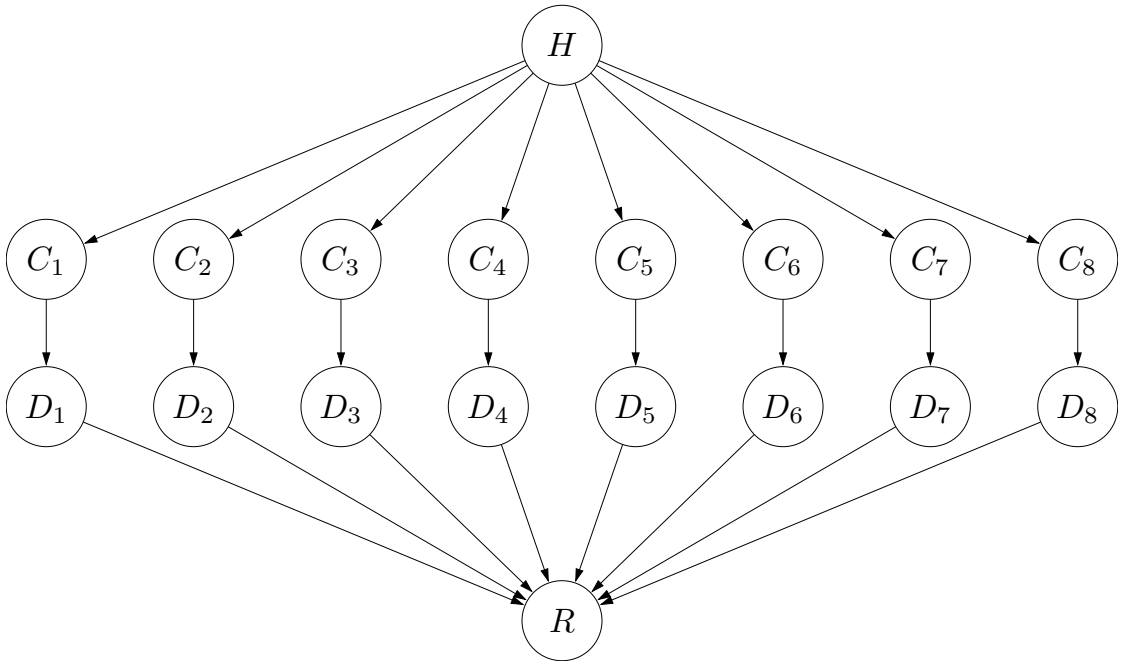


Figure 2: Bayesian network 4 25 corresponding to the voting process in Bayesland.

a. (*10 points*)

You suspect that the voting process is actually much simpler than it appears if you could only expose some of the conditional independence assumptions. Mark each of the following statements as either true or false.

i	$C_2 \perp\!\!\!\perp C_4$	True / False
ii	$C_2 \perp\!\!\!\perp C_4 \mid H$	True / False
iii	$C_2 \perp\!\!\!\perp C_4 \mid R$	True / False
iv	$C_2 \perp\!\!\!\perp C_4 \mid H, R$	True / False
v	$C_2 \perp\!\!\!\perp C_4 \mid D_2, D_4$	True / False
vi	$C_2 \perp\!\!\!\perp C_4 \mid D_2, D_4, R$	True / False
vii	$C_2 \perp\!\!\!\perp C_4 \mid H, D_2$	True / False
viii	$C_2 \perp\!\!\!\perp C_4 \mid H, D_2, R$	True / False
ix	$C_2 \perp\!\!\!\perp C_4 \mid H, D_2, D_4$	True / False
x	$C_2 \perp\!\!\!\perp C_4 \mid H, D_2, D_4, R$	True / False

Solution To ensure d-separation on all paths betwen C_2 and C_4, both the following conditions must hold:

- H is is conditioned on.
- R is not being conditioned on (v-structure), or at least one of D_2 and D_4 are being conditioned on.

From the above, we get

i False

ii True

iii False

iv False

v False

vi False

vii True

viii True

ix True

x True

b. (*10 points*)

After having discovered the qualitative structure of the voting process, you are now interested in digging a bit deeper into how various people influence each other. Compute the following (write each expression in terms of α and β). Hint: for each query, try to marginalize away all the irrelevant quantities before you start writing down formulas. If things get hairy, you might be doing something wrong!

i What did the head say given the citizens' votes?

$\mathbb{P}(H = 1 \mid C_1 = 1, C_2 = 1, C_3 = 0) =$

Solution The relevant factors are $p(h)$, $p(c_1 \mid h)$, $p(c_2 \mid h)$, $p(c_3 \mid h)$. Plugging the desired values, we get that for $H = 1$, we have a weight of $(1-\alpha)^2\alpha$ and for $H = 0$, we have a weight of $\alpha^2(1-\alpha)$. Normalizing, we get

$$\mathbb{P}(H = 1 \mid C_1 = 1, C_2 = 1, C_3 = 0) = \frac{(1-\alpha)^2\alpha}{(1-\alpha)^2\alpha + (1-\alpha)\alpha^2} = 1 - \alpha.$$

---

ii How does knowing one citizen's vote influence our belief of another citizen's vote?

$\mathbb{P}(C_5 = 1 \mid C_8 = 1) =$

Solution All variables other than H, C_5, C_8 can be marginalized out as a leaf, so we are left with the relevant factors $p(h)$, $p(c_5 \mid h)$ and $p(c_8 \mid h)$. Marginalizing out H, we get a new factor $f(c_5, c_8) = \sum_h p(h)p(c_5 \mid h)p(c_8 \mid h)$. Condition on $C_8 = 1$.

- For $C_5 = 1$, we have $f(c_5, c_8) = \frac{1}{2}(1-\alpha)^2 + \frac{1}{2}\alpha^2$.
- For $C_5 = 0$, we have $f(c_5, c_8) = \frac{1}{2}\alpha(1-\alpha) + \frac{1}{2}(1-\alpha)\alpha$.

Normalizing, we get:

$$\mathbb{P}(C_5 = 1 \mid C_8 = 1) = (1-\alpha)^2 + \alpha^2 = 1 - 2\alpha + 2\alpha^2. \tag{18}$$

---

iii You're getting tired of computing all these queries by hand, so you decide to use Gibbs sampling. Describe the Gibbs sampling update for D_i as a function of β. You must simplify as much as possible and explain what the resulting conditional distribution of D_i is.

Solution The Gibbs update is to sample from D_i according to

$$\mathbb{P}(D_i \mid C_i, R, D_{-i}) \propto p(d_i \mid c_i) p(r \mid d_1, \ldots, d_n),$$

where $p(r \mid d_1, \ldots, d_n) = [r = [\sum_{i=1}^{n} d_i > n/2]]$, which is either 0 or 1. There are two possible cases:

- If both values of $D_i \in \{0, 1\}$ lead to $p(r \mid d_1, \ldots, d_n) = 1$, then we sample D_i only from $p(d_i \mid c_i)$, which is setting D_i to C_i with probability $1 - \beta$ and $1 - C_i$ with probability β.
- Otherwise, there is only one possible value that D_i can take, so the Gibbs sampler must fix D_i to the same value.

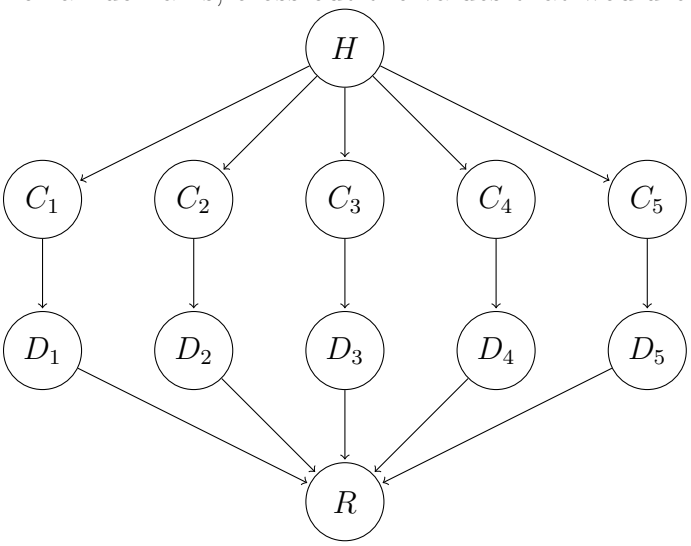
c. (*10 points*)

You're still trying to make your Gibbs sampling go fast. You remember that a Bayesian network is an instance of a factor graph, so you decide to apply AC-3.

Specifically, suppose there are $n = 5$ citizens and $\beta = 0$ (no noise in transferring votes). Consider the factor graph corresponding to

$$\mathbb{P}(H, C_1, C_2, C_3, C_4, C_5, D_3, D_4, D_5 \mid D_1 = 0, D_2 = 0, R = 1).$$

For all domains, cross out the values that would be pruned by running AC-3:



1. $H \in \{0, 1\}$
2. $C_1 \in \{0, 1\}$
3. $C_2 \in \{0, 1\}$
4. $C_3 \in \{0, 1\}$
5. $C_4 \in \{0, 1\}$
6. $C_5 \in \{0, 1\}$
7. $D_3 \in \{0, 1\}$
8. $D_4 \in \{0, 1\}$
9. $D_5 \in \{0, 1\}$

Solution Since $D_1 = D_2 = 0$ and $R = 1$, this means $D_3 = D_4 = D_5 = 1$. Next, $\beta = 0$ means that $C_i = D_i$ for all i, so all the C_i's are fixed to the D_i's. Thus, all variables are clamped except H.

d. (*10 points*)

Again assume $\beta = 0$ and suppose you're interested in how citizens voted given that the overall outcome was positive:

$$\mathbb{P}(C_1, \ldots, C_n \mid R = 1).$$

Suppose you want to use particle filtering for this task. In class, we defined particle filtering for an HMM with transitions and emissions, so we'll need to generalize slightly.

(i) Define a set of new variables $E_1, \ldots, E_n$, where $E_i = 1$ if it is possible for $R = 1$ based on $C_1, \ldots, C_i$ and $E_i = 0$ otherwise. The upshot is that conditioning on $E_1, \ldots, E_n$ is the same as conditioning on $R = 1$, but the evidence can be partially evaluated without seeing all of $C_1, \ldots, C_n$. Define the local conditional distribution formally:

Solution We know that it's not possible for $R = 1$ if the total number of negative votes exceeds $n/2$:

$$p(e_i = 0 \mid c_1, \ldots, c_i) = \left[\sum_{j=1}^{i} (1 - c_j) \geq n/2\right].$$

Put positively:

$$p(e_i = 1 \mid c_1, \ldots, c_i) = \left[\sum_{j=1}^{i} c_j > i - n/2\right].$$

$p(e_i \mid c_1, \ldots, c_i) =$ _____

(ii) In an HMM, we proposed extensions to a particle $(c_1, \ldots, c_{i-1})$ by sampling C_i from $p(c_i \mid c_{i-1})$. For an non-HMM, we need to sample C_i conditioned on the entire history. Compute this quantity (Your answer does not have to be in terms of α and β, but it must be in terms of known probabilities):

Solution

$$\mathbb{P}(C_i = c_i \mid C_1 = c_1, \ldots, C_{i-1} = c_{i-1}) \propto \sum_h p(h) \prod_{j=1}^{i} p(c_j \mid h).$$

$\mathbb{P}(C_i = c_i \mid C_1 = c_1, \ldots, C_{i-1} = c_{i-1}) \propto$ _____

e. (*10 points*)

Having done all these calculations, you realize that you don't actually know what α and β actually are! So you decide to record all the communications in Bayesland in order to estimate these values.

Suppose we have $n = 5$ citizens, and they voted on two issues. For the first issue, the head of state chose $H = 1$, and the following results were received (and the result was $R = 1$):

i	1	2	3	4	5
C_i	1	0	1	1	0
D_i	0	1	1	1	0

For the second issue, the head of state chose $H = 0$, and the following results were received (and the result was $R = 0$):

i	1	2	3	4	5
C_i	0	0	0	1	1
D_i	0	0	0	1	1

Taking all this data into account, what is the maximum likelihood estimate of α and β with add λ smoothing / laplace smoothing? Your answer should be in terms of λ.

Solution By simple counting, we get $\alpha = \frac{(0+1+0+0+1)+(0+0+0+1+1)}{10} = \frac{4+\lambda}{10+2\lambda}$.

$\alpha =$ _____

Solution By simple counting, we get $\beta = \frac{(1+1+0+0+0)+(0+0+0+0+0)}{10} = \frac{2+\lambda}{10+2\lambda}$.

$\beta =$ _____