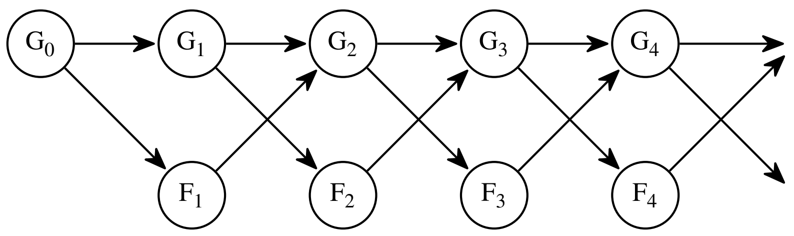
CS 188 Fall 2024 Introduction to Artificial Intelligence

Final Review Solutions

Q1. Dynamic Bayes' Nets: Book Club

Each week t ($t \geq 0$), the members of a book club decide on the genre of the book G_t to read that week: romance ($+g_t$) or science fiction ($-g_t$). Each week's choice is influenced by the genre chosen in the previous week. In addition, after week t ($t \geq 0$), they collect feedback (F_{t+1}) from the members, which can be overall positive ($+f_{t+1}$) or negative ($-f_{t+1}$), and that influences the choice at week $t+2$. The situation is shown in the Bayes' net below:



(a) Recall that the joint distribution up to time T for the "standard" HMM model with state X_t and evidence E_t is given by $P(X_{0:T}, E_{1:T}) = P(X_0)\prod_{t=1}^{T} P(X_t \mid X_{t-1})P(E_t \mid X_t)$.

Write an expression for the joint distribution in the model shown above.

$P(F_{1:T}, G_{0:T}) =$ $\boxed{P(G_0)P(F_1 \mid G_0)P(G_1 \mid G_0)\prod_{t=2}^{T} P(F_t \mid G_{t-1})P(G_t \mid G_{t-1}F_{t-1})}$

This is just the ordinary expression for the joint distribution of a Bayes net. Recall that in a Bayes net, multiplying all of the conditional probability tables together (one per node) results in the joint distribution. The tricky part to writing this expression is getting the first couple of steps right ($t = 0$ and $t = 1$).

(b) Which of the following Markov assumptions are implied by the network structure, assuming $t \geq 2$?

- ☐ $P(G_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t \mid F_{t-1})$
- ☐ $P(G_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t \mid G_{t-1})$
- ☐ $P(G_t, F_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t, F_t \mid G_{t-1})$
- [✓] $P(G_t, F_t \mid G_{0:t-1}, F_{1:t-1}) = P(G_t, F_t \mid G_{t-1}, F_{t-1})$
- ☐ $P(F_t \mid G_{0:t-1}, F_{1:t-1}) = P(F_t \mid F_{t-1})$
- [✓] $P(F_t \mid G_{0:t-1}, F_{1:t-1}) = P(F_t \mid G_{t-1}, F_{t-2})$
- ○ None of the above

These can all be worked out precisely using independence of non-descendants given parents. For the last one, note that the F_{t-2} is superfluous, but the equation is still correct.

The conditional probability tables for G_t and F_t are shown below, where a, b, c, d, p, q are constants between 0 and 1.

F_t	G_{t-1}	$P(F_t \mid G_{t-1})$
+	+	p
−	+	$1-p$
+	−	q
−	−	$1-q$

G_t	F_{t-1}	G_{t-1}	$P(G_t \mid F_{t-1}, G_{t-1})$
+	+	+	a
−	+	+	$1-a$
+	+	−	b
−	+	−	$1-b$
+	−	+	c
−	−	+	$1-c$
+	−	−	d
−	−	−	$1-d$

Let's consider this model as a Markov chain with state variables (F_t, G_t).

(c) Write out the transition probabilities below.

Your answer should be an expression, possibly in terms of a, b, c, d, p, and q.

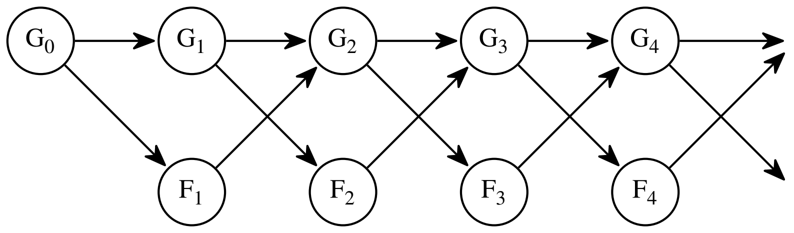
$x_1 = P(+f_t, +g_t \mid +f_{t-1}, +g_{t-1}) = \boxed{ap}$ $x_2 = P(+f_t, +g_t \mid +f_{t-1}, -g_{t-1}) = \boxed{bq}$

$x_3 = P(+f_t, +g_t \mid -f_{t-1}, +g_{t-1}) = \boxed{cp}$ $x_4 = P(+f_t, +g_t \mid -f_{t-1}, -g_{t-1}) = \boxed{dq}$

These entries follow from the fact that $P(F_t, G_t \mid F_{t-1}, G_{t-1}) = P(F_t \mid G_{t-1})P(G_t \mid F_{t-1}, G_{t-1})$. You can think of this expression as joining together two factors (CPTs) of the Bayes' net.

Intuitively, you can also derive these expressions by considering the transition dynamics of this Markov chain. For example, consider deriving x_1. At time $t-1$, we have state $+f_{t-1}, +g_{t-1}$, and we want to know how likely it is for the state at time t to be $+f_t, +g_t$. In order for this transition to happen, $+f_{t-1}$ must transition to $+f_t$. The probability that this happens is governed by $P(F_t|G_{t-1})$, and we look up the value in the table $P(+f_t| + g_{t-1})$ (since we know $+g_{t-1}$ at time $t-1$) to find p. Then, $+g_{t-1}$ must transition to $+g_t$. This transition is governed by the $P(G_t|F_{t-1}, G_{t-1})$, and substituting in the desired values $P(+g_t| + f_{t-1}, +g_{t-1})$ lets us look up the value a in the table. Since both of these transitions must happen, the probability of the overall transition is ap. The same approach can be used to derive the other three expressions.

The Bayes' net, repeated for your convenience:



(d) As t goes to infinity, the stationary distribution of this Markov chain is shown below, where y_1, y_2, y_3, y_4 are constants between 0 and 1.

$$y_1 = P(+f_\infty, +g_\infty)$$
$$y_2 = P(+f_\infty, -g_\infty)$$
$$y_3 = P(-f_\infty, +g_\infty)$$
$$y_4 = P(-f_\infty, -g_\infty)$$

Write an equation that must be true if this is a stationary distribution.

Your answer should be an equation, possibly in terms of x_1, x_2, x_3, x_4 (from the previous part), y_1, y_2, y_3, y_4.

$$\boxed{x_1y_1 + x_2y_2 + x_3y_3 + x_4y_4 = y_1}$$

Recall that in the equilibrium distribution, the probability of being in a state no longer depends on the time step. In other words, if the probabilities of being in the four states at time ∞ are y_1, y_2, y_3, y_4, then at the next time step $\infty + 1$, after the transition dynamics are applied once, the probabilities of being in the four states are still y_1, y_2, y_3, y_4.

From the previous subpart, we have the probabilities of transitioning into state $(+f, +g)$ from each of the four possible states. Therefore, we can write an equilibrium equation about the probability of being in $(+f, +g)$, which is y_1.

$x_1 y_1 =$ probability of starting in $(+f, +g)$, then transitioning to $(+f, +g)$. Note that x_1 is the probability of the transition and y_1 is the probability of starting in the state.

$x_2 y_2 =$ probability of starting in $(+f, -g)$ and transitioning to $(+f, +g)$.

$x_3 y_3 =$ probability of starting in $(-f, +g)$ and transitioning to $(+f, +g)$.

$x_4 y_4 =$ probability of starting in $(-f, -g)$ and transitioning to $(+f, +g)$.

Since these are the only four ways to transition into $(+f, +g)$, if we sum them up, we should get the probability that we're in $(+f, +g)$ on the next time step, and because this is an equilibrium distribution, this should be equal to the probability that we're in $(+f, +g)$ right now.

Using the matrix–vector form of the Markov chain update: $\mathbf{T}^\top \pi = \pi$, where $\mathbf{T}$ is the transition matrix and $\pi = (y_1, y_2, y_3, y_4)^\top$ is the stationary distribution. The probabilities X_1, x_2, x_3, x_4 are the first column of $\mathbf{T}$ and hence the first row of $\mathbf{T}^\top$, so the solution is just the first line of the matrix–vector equation.

(e) Select all true statements.

[✓] There are values of a, b, c, d such that the feedback model defined by p, q is irrelevant to the stationary distribution of the genre G.

☐ Even if $P(F_t, G_t)$ reaches a unique stationary distribution, it is possible that the marginal probabilities $P(F_t)$ and $P(G_t)$ do not reach unique stationary distributions.

☐ For any values of a, b, c, d, p, q, there is at least one stationary distribution.

○ None of the above

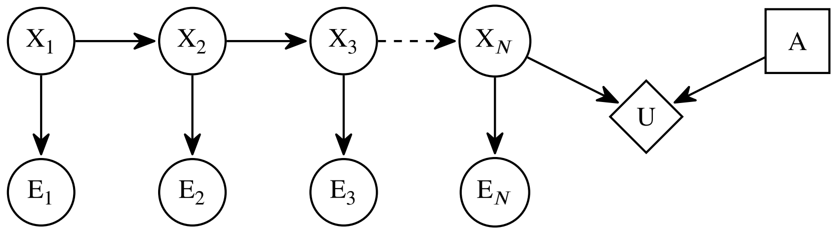
1. True. If $a = c = 1$ and $b = d = 0$, then G_t is simply copied from G_{t-1} regardless of F. In that case the stationary distribution is the same as $P(G_0)$, and p and q are irrelevant for computing the stationary distribution.

2. False. $P(F_t)$ and $P(G_t)$ can be computed directly from $P(F_t, G_t)$ by summing out, so if the latter is constant then so are the former.

3. False. If $a = c = 0$ and $b = d = 1$, then G_t is always the opposite of G_{t-1}, i.e., a deterministic cycle, and there is no stationary distribution.

Q2. HMMs and VPI: Markov Menagerie

Consider the following hidden Markov model (HMM). All random variables are binary.



We would like to compute $\text{VPI}(E_N) = \text{MEU}(E_N) - \text{MEU}(\emptyset)$.

(a) Which of these distributions is needed to compute $\text{MEU}(\emptyset)$?

[✓ $P(X_N)$] $P(X_N|X_{N-1})$ $P(X_N|E_1,\dots,E_N)$ $P(X_N|E_N)$

The utility U depends on X_N, so we need a distribution over X_N to compute the expected utility of each action. $\text{MEU}(\emptyset)$ means that we are given no evidence, so the distribution over X_N should also be given no evidence.

(b) Which of these computations can be used to derive the distribution in part **(a)**?

- Run the forward algorithm with no modifications.
- Run the forward algorithm, skipping all time elapse updates.
- [✓ Run the forward algorithm, skipping all observation updates.]
- Read the conditional probability table under X_N.

We need $P(X_N)$. This is not in the Bayes' net; the CPT under X_N is $P(X_N|X_{N-1})$.
To obtain $P(X_N)$, we need to run the forward algorithm, skipping all observation updates, because we have no evidence.

(c) Write an expression that can be used to compute $\text{MEU}(E_N)$.

$$\text{MEU}(E_N) = \sum_{e_N} \textbf{(i)} \left[\max_a \sum_{x_N} \textbf{(ii)}\ \textbf{(iii)} \right]$$

(i)	[✓ $P(E_N)$]	$P(E_N\|E_{N-1})$	$P(E_N\|X_N)$	$P(E_N\|X_1,\dots,X_N)$
(ii)	$P(X_N\|X_{N-1})$	$P(X_N\|X_1,\dots,X_{N-1})$	[✓ $P(X_N\|E_N)$]	$P(X_N\|E_1,\dots,E_N)$
(iii)	1	$U(x_N)$	$U(a)$	[✓ $U(x_N,a)$]

$$\sum_{e_N} P(e_N) \left[\max_a \sum_{x_N} P(x_N|e_N) U(x_N,a) \right]$$

We need $P(X_N|E_N)$ to compute the expected utility of each action.
Then, we also need $P(E_N)$ so that we can weight each expected utility by the probability of that specific evidence occurring.
Finally, we need the utility, which depends on both the value of X_N and the action selected.

(d) Consider the distribution in part **(a)**, which you computed in part **(b)**.
From the distribution in **(a)**, which of the following additional computations results in the distribution in blank **(ii)**?

- [✓ Apply an additional observation update.]
- ○ Apply an additional time elapse update.
- ○ Apply an additional time elapse and observation update.
- ○ Re-run the forward algorithm from the beginning, with no modifications.

From the previous subpart, we have $P(X_N)$, but we need $P(X_N|E_N)$. This requires one extra evidence update in the forward algorithm: we need to weight every value in the table $P(X_N)$ by $P(E_N|X_N)$, and then normalize.

In equations: We have $P(X_N)$ (from the earlier subparts) and $P(E_N|X_N)$ (from the Bayes' net). We want $P(X_N|E_N)$, so we can apply Bayes' rule:

$$P(X_N|E_N) = \frac{P(X_N)P(E_N|X_N)}{P(E_N)}$$

In other words, in the numerator, we're multiplying every value in the $P(X_N)$ table by $P(E_N|X_N)$, and then we normalize (since the denominator is a constant).

(e) Which of the following computations can be used to derive the distribution in blank **(i)**?

- ○ $P(x_N)P(E_N|x_N)$
- [✓ $\sum_{x_N} P(x_N)P(E_N|x_N)$]
- ○ $\sum_{e_N} P(x_N)P(e_N|x_N)$
- ○ $\sum_{x_N} P(E_N)P(x_N|E_N)$

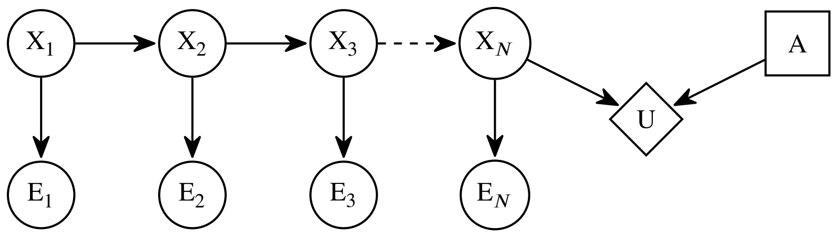
We're given $P(X_N)$ (from earlier subparts), and $P(E_N|X_N)$ (from the Bayes' net). Using the chain rule, we can get:

$$P(X_N)P(E_N|X_N) = P(X_N, E_N)$$

Then, you can sum out X_N to get:

$$\sum_{x_N} P(x_N)P(E_N|x_N) = P(E_N)$$

The diagram, repeated for your convenience:



For the rest of the question, suppose we would like to compute $\text{VPI}(E_1, \ldots, E_N) = \text{MEU}(E_1, \ldots, E_N) - \text{MEU}(\emptyset)$.

(f) To compute $\text{MEU}(E_1, \ldots, E_N)$, we'll need one or more distribution(s) over X_N.
Which of these computations will generate the necessary distribution(s) over X_N?

- ○ Run the forward algorithm once, with no modifications.
- [✓ ● Run the forward algorithm 2^N times, with no modifications.]
- ○ Run the forward algorithm once, skipping all observation updates.
- ○ Run the forward algorithm 2^N times, skipping all observation updates.

We don't know what the evidence is, so we have to run the forward algorithm once for every possible setting of the evidence $P(X_N|e_1, \ldots, e_N)$.

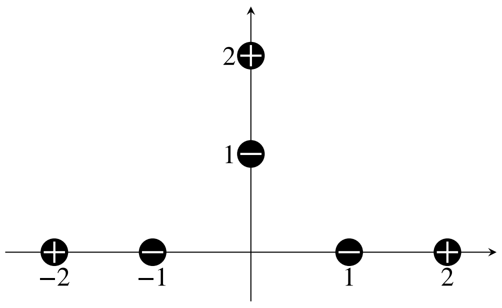
(g) To compute $\text{MEU}(E_1, \ldots, E_N)$, which other distribution do we need?

○ $P(E_1)$ ○ $P(E_N)$ [✓ ● $P(E_1, \ldots, E_N)$] ○ $P(E_1, \ldots, E_N|X_N)$

We need to weight each $\text{MEU}(e_1, \ldots, e_N)$ by the corresponding probability of evidence, $P(E_1, \ldots, E_N)$.

Q3. Higher-Dimensional Perceptrons

Consider a dataset with 6 points on a 2D coordinate grid. Each point belongs to one of two classes. The points $[-1, 0], [1, 0], [0, 1]$ belong to the negative class. The points $[-2, 0], [2, 0], [0, 2]$ belong to the positive class.



(a) Suppose we run the perceptron algorithm with the initial weight vector set to $[0, 5]$.

What is the updated weight vector after processing the data point $[0, 1]$?

$\boxed{[0, 4]}$

We have $f = [0, 1]$ and $w = [0, 5]$. First we classify the point by computing the dot product: $f \cdot w = 5$. This classifies f in the positive class, but the true class is negative.

Our classification is wrong, so we need to adjust the weights by subtracting the feature vector: $w - f = [0, 5] - [0, 1] = [0, 4]$.

(b) How many iterations of the perceptron algorithm will run before the algorithm converges? Processing one data point counts as one iteration. If the algorithm never converges, write ∞.

$\boxed{\infty}$

The data points are not linearly separable, so the perceptron algorithm will never terminate.

In the next few subparts, we'll consider *transforming* the data points by applying some modification to each of the data points. Then, we pass these modified data points into the perceptron algorithm.

For example, consider the transformation $[x, y] \to [x, y, x^2, 1]$. In this transformation, we add two extra dimensions: one whose value is always the square of the first coordinate, and one whose value is always the constant 1. For example, the point at $[2, 0]$ is transformed into a point at $[2, 0, 4, 1]$ in 4-dimensional space.

(c) Which of the following data transformations will cause the perceptron algorithm to converge, when run on the transformed data? Select all that apply.

- ☐ $[x, y] \to [y, x]$
- ☐ $[x, y] \to [x, y, 1]$
- ■ $[x, y] \to [x, y, x^2, 1]$
- ☐ $[x, y] \to [x, y, x^2 + y^2]$
- ○ None of the above.

(A): False. Pictorially, this transformation reflects all the data points across the $x = y$ line. If you graph the resulting points, they're still not linearly separable, so the perceptron algorithm still never terminates.

(B): Pictorially, this transformation plots the data points along a flat plane in the 3D grid. If you graph the resulting points, they're still not linearly separable, so the perceptron algorithm still never terminates.

If you don't want to picture points in higher dimensions, another solution is to note that the resulting decision boundary here is $w_1 x + w_2 y + w_3 > 0$. In other words, you added a y-intercept term w_3 to the decision boundary line on the 2D coordinate plane. Adding a y-intercept so that the decision boundary doesn't have to cross the origin still doesn't help us linearly separate the data, though.

(C): True. Write the decision boundary equation $w_1x+w_2y+w_3x^2+w_4 > 0$, and note that this is some form of parabola (quadratic equation) in the 2D coordinate plane, since we have terms with y, x^2, x, plus some constant.

Intuitively, you could sketch a parabola that crosses coordinate points $[-1.5, 0]$, $[0, 1.5]$, and $[1.5, 0]$ on the coordinate grid, which would separate the points.

If you wanted to find the exact equation of this line (which was not necessary for this question), you could start with $y = x^2$. Then note that the parabola has to point down, so it should be something like $y = -x^2$. Then note that we should shift this parabola upwards so that the negative points are "below" the parabola, to get something like $y = -x^2 + 1.5$. Rearranging gives the decision boundary $y + x^2 - 1.5 > 0$.

You can confirm that this equation works by plugging in all six points and noting that the negative points all have $y+x^2-1.5 < 0$, and all the positive points have $y + x^2 - 1.5 > 0$.

(D): False. The new feature, $x^2 + y^2$, is equal to 1 for all the negative points and 4 for all the positive points. However, the perceptron classifies points based on the sign of the output, so we'd have somehow use the remaining features to map all the 1s to negative numbers, and all the 4s to positive numbers. We don't have a constant (like in the previous options) to help us with this, and trying to add/subtract multiples of x or y proves to not be useful either (playing around with a few possibilities should be enough to convince you that x and y can't help here).

A geometric solution (which is not necessary to solve this problem, but useful if you like thinking geometrically): note that the x^2+y^2 term looks like the equation of a circle, so adding this term introduces circles into our decision boundaries. We can draw circles and classify points inside the circle as one class, and points outside the circle as a different class. However, we lack a constant term, so our decision boundary is always going to be in the form $w_1x+w_2y+w_3(x^2+y^2) > 0$. We can see that $[0, 0]$ is always going to fall on the decision boundary, so the lack of constant term restricts our decision boundaries to only the circles that pass through the origin. Visually, we can see that a circle passing through the origin will not separate the points.

(d) Suppose we transform $[x, y]$ to $[x, y, x^2 + y^2, 1]$, and pass the transformed data points into the perceptron.

Write one possible weight vector that the perceptron algorithm may converge to.

[0,0,1,-2]

The new x^2+y^2 coordinate looks very useful for separating the data. Note that for the negative points, the new coordinate is always 1, and for the positive points, the new coordinate is always 4.

However, 1 and 4 are both positive, and the perceptron classifies based on the sign of the output. To fix this, we need to use the constant bias term to add any negative bias $-4 < c < -1$ from every classification so that 1 maps to some negative number and 4 maps to some positive number. For example, $c = -2$ would map positive points to $1-2 = -1$ and negative points to $4-2 = 2$.

If you used a different weight k for the x^2+y^2 feature, you would get perceptron activation of k for all the negative points and $4k$ for all the positive points. Then, your constant factor would need to be in the range $-4k < c < -k$ so that k gets mapped to a negative number, and $4k$ gets mapped to a positive number.

A nice geometric solution (which is not necessary to solve this problem): the x^2+y^2 feature, along with the constant bias, lets us draw circular decision boundaries. The restriction to circles passing through the origin, from the previous subpart, no longer applies here because we've introduced a constant bias. Now, we can draw a circle that separates the points. Any circle with center at the origin, and radius between 1 and 2, will perfectly separate the points (all negative points inside, all positive points outside). If we set the radius to be 1.5, we'd get the circle equation $x^2+y^2 = 1.5^2 = 2.25$. Rearranging terms a bit, we get the decision boundary $x^2+y^2-2.25 > 0$. This corresponds to the weight vector $[0, 0, 1, -2.25]$, which is also a correct solution.

(e) Construct another transformation (not equal to the ones above) that will allow the perceptron algorithm to converge.

Hint: The transformation $[x, y] \to [x, y, x^2 + y^2, 1]$ allows the perceptron algorithm to converge.

Fill in the blank: $[x, y] \to [x, y, \_\_\_\_, 1]$.

$x^4 + y^4$

One simple class of transformations that works here is any that combines the magnitudes of the two coordinates. (Pictorially, this corresponds to the fact that the negative points are closer to the origin, and the positive points are further away from the origin.) Some sample answers include: $x^4 + y^4$, or $x^6 + y^6$, or $|x| + |y|$, etc.

Another simple class of transformations is to add a constant factor to the x^2+y^2 feature that helped from earlier: $2(x^2+y^2)$, or $3(x^2 + y^2)$, or $4(x^2 + y^2)$, etc. These transformations still work because you could always adjust the third weight value from the $[x, y, x^2 + y^2, 1]$ perceptron to cancel out the new coefficient, which would give you back the original decision boundary that worked on the $[x, y, x^2 + y^2, 1]$ perceptron. For example, if your transformation is $x^2 + y^2 \rightarrow c(x^2 + y^2)$ and the original weight vector had w_3, you could adjust the weight vector to w_3/c and end up with the same decision boundary.

Another simple class of transformation is to add a constant value to the x^2+y^2 feature from earlier: x^2+y^2+1, x^2+y^2+2, etc. If your transformation is $x^2 + y^2 \rightarrow x^2 + y^2 + c$, then you've added a constant value cw_3 to every activation value. If you adjust the constant bias weight value from w_4 to $w_4 - cw_3$, then you cancel out the new addition and end up with the same original decision boundary.

Other solutions probably exist here, but these were the simplest three that we could think of.

Exam continues on next page.

Q4. ML: Spam Filter

Pacman has hired you to work on his PacMail email service. You have been given the task of designing a spam detector.

You are given a dataset of emails X, each with labels Y of "spam" or "ham." Here are some examples from the dataset.

Spam Email Ex. 1: "WINNER!! As a valued network customer you have been selected to receive a $900 prize reward!!!"

Spam Email Ex. 2: "We are trying to contact you. Last weekend's draw shows that you won a £1000 prize GUARANTEED!!!"

Ham Email Ex. 1: "Hey! Did you want to grab coffee before the team meeting on Friday?"

Ham Email Ex. 2: "Thank you for attending the talk this morning. I've attached the presentation for you to share with your team. Please let me know if you have any questions."

Your job is to classify the emails in a second dataset, the test dataset, which do not have labels.

(a) Considering only the examples given, which of the following features, in isolation, would be sufficient to classify the examples correctly using a linear classifier?

- ☐ The number of words in the email.
- ■ The number of times the exclamation point ("!") appears in the email.
- ■ The number of times "prize" appears in the email.
- ■ The number of capital letters in the email.
- ○ None of the above.

For all options, the question asks us to use a linear classifier with a single feature, which means that our classifier must have a linear boundary such that all items on one side are spam, and all items on the other side are ham. In a one-dimensional space (one feature), the linear boundary would be a single point on the number line, where all items to the left (less than the boundary number) are one class, and all items to the right (greater than the boundary number) are the other class.

Option 1: False. The two spam emails have 16 and 17 words, and the two ham emails have 13 and 28 words. We cannot draw a linear boundary in this case. There is no boundary number such that 16 and 17 are greater, and 13 and 28 are less (or vice-versa).

Option 2: True. The two spam emails have 5 and 3 exclamation points, and the two ham emails have 1 and 0 exclamation points. We can set a boundary at 2, and classify all emails with more than 2 exclamation points as spam, and all emails with less than 2 exclamation points as ham. (Other boundaries also exist, e.g. 1.5 exclamation points.)

Option 3: "Prize" appears once in each of the spam emails, and zero times in each of the ham emails. We can set a decision boundary at 0.5, and classify all emails with more than 0.5 "prize" appearances as spam, and all emails with less than 0.5 "prize" appearances as ham.

Option 4: The two spam emails have 7 and 12 capital letters, and the two ham emails have 3 and 3 capital letters. We can set a decision boundary at 6, and classify all emails with more than 6 capital letters as spam, and all emails with less than 6 capital letters as ham.

(b) Select all true statements about using naive Bayes to solve this problem.

- ■ We assume that each feature is conditionally independent of the other features, given the label.
- ■ Given that the prior (class) probabilities are the same, the probability of classifying an email as "spam", given the contents of the email, is proportional to the probability of the contents, given that the email is labeled "spam".
- ☐ Including more features in the model will always increase the test accuracy.
- ■ Naive Bayes uses the maximum likelihood estimate to compute probabilities in the Bayes net.
- ○ None of the above.

Option 1 is true. By definition, the Naive Bayes' model assumes that features are conditionally independent, given the label. This is a result of the Bayes' Net structure, where all features are direct descendants of the label. Running d-separation on the Bayes' net will show that all features must be conditionally independent given the label.

Option 2 is true. Recall that in Naive Bayes', the probability of classifying an email as spam, given features $f_1, \dots, f_n$, is:

$$P(Y = \text{spam}|f_1, \dots, f_n) = \frac{P(\text{spam}) \cdot \prod_{i=0}^{n} P(f_i|\text{spam})}{P(\text{spam}) \cdot \prod_{i=0}^{n} P(f_i|\text{spam}) + P(\text{ham}) \cdot \prod_{i=0}^{n} P(f_i|\text{ham})}$$

However, if the prior probabilities are the same, i.e. $P(\text{spam}) = P(\text{ham})$, then we can cancel terms and get:

$$P(Y = \text{spam}|f_1, \dots, f_n) = \frac{\prod_{i=0}^{n} P(f_i|\text{spam})}{\prod_{i=0}^{n} P(f_i|\text{spam}) + \prod_{i=0}^{n} P(f_i|\text{ham})}$$

The probability of the contents, given that the email is labeled spam, is:

$$P(f_1, \dots, f_n|Y = \text{spam}) = \prod_{i=0}^{n} P(f_i|\text{spam})$$

These two values are proportional, since they only differ in the additional denominator in the first value.

Option 3 is false. Intuitively, it's possible that we include features with no useful predictive power, which will not help us classify the test dataset (which we've never seen before).

Option 4 is true. The probabilities in the underlying Bayes' net are computed using the count estimate, which is the maximum likelihood estimate, as seen in lecture. For example, under the label node in the Bayes' net, we estimate $P(Y = \text{spam})$ and $P(Y = \text{ham})$, the prior probabilities, by computing the count estimate of how many spam and ham emails are in the training dataset. This is the maximum likelihood estimate of the true prior probabilities.

(c) You want to determine whether naive Bayes or logistic regression is better for your problem. Select all true statements about these two methods.

- ■ Logistic regression requires fewer learnable parameters than naive Bayes, assuming the same features.
- □ Both logistic regression and naive Bayes use the same independence assumption.
- ■ Both logistic regression and naive Bayes can be used for multi-class classification.
- ■ Logistic regression models the conditional class distribution $P(Y|W)$ directly, whereas naive Bayes models the joint distribution $P(Y, W)$.
- ○ None of the above.

Option 1 is true. In logistic regression (perceptron with non-linearity at the end to output probabilistic decision), we need one parameter per feature. In Naive Bayes', we need at least one parameter per feature (e.g. if feaures are binary, each feature node in the Bayes' net has one parameter), plus at least one more parameter for the prior probabilites (corresponding to the label node in the Bayes' net). Therefore, Naive Bayes' will require more learnable parameters.

Option 2 is false. Logistic regression does not make the assumption that the features are conditionally independent, given the label.

Option 3: True. Naive Bayes' can be extended to multiple classes by adding more entries in the underlying conditional probability tables. For example, the label node could now have $P(Y = \text{spam})$, $P(Y = \text{ham})$, and $P(Y = \text{other})$. Also, the feature nodes could have $P(F_i|Y)$ for all values of Y (even if there are more than 2 values for the label Y).

Logistic regression can be extended to multiple classes, similar to how perceptrons can be extended to multi-class classification problems. We can compute activations for each class, and then use the softmax to convert the activations to probabilities.

Option 4: True. In logistic regression, we turn the activations directly into probabilities $P(Y|W)$. However, in Naive Bayes', we use the Bayes' net to model the joint distribution $P(Y, W)$, and then we normalize at the end to convert the joint distribution to the conditional probabilities $P(Y|W)$.

You decide to use binary bag-of-words (see definition below) to extract a feature vector from each email in the dataset.

Binary bag-of-words: given a vocabulary of N words, bag-of-words represents a string as an N–element vector, where the value at index i is 1 if word i appears in the string, and 0 otherwise.

The next 3 subparts (d)-(f) are connected. After training a naive Bayes model with **binary bag-of-words** features, you compute the following probability tables for $P(W = w_i | Y = y)$, where w_i is the ith word in your vocabulary. Assume that there are no other words in your model's vocabulary.

	"hey"	"valued"	"team"	"share"
Spam	0.25	0.6	0.3	0.8
Ham	0.4	0.1	0.5	0.3

Now you are tasked with classifying this new email:

"Hey, what time is our team meeting? Can't wait to share with the team!!!"

(d) Fill in the following table with integers corresponding to the bag-of-words vector **w** for the new email. Ignore punctuation and capitalization.

	"hey"	"valued"	"team"	"share"
w	1	0	1	1

"hey", "team", and "share" all appear in the data point, so according to binary bag-of-words, the elements in the feature vector corresponding to these words should be assigned the value 1.

"valued" does not appear in the data point, so the element in the feature vector corresponding to this word should be assigned the value 0.

Note that as we are using binary bag-of-words, "team" has a corresponding value of 1, even though the word appears multiple times in the data point.

(e) Compute the probability distribution $P(Y, W = \mathbf{w})$ for this email. You may assume the prior probabilities of each class are equal, i.e. $P(Y = \text{spam}) = P(Y = \text{ham}) = 0.5$. You may ignore words in the sentence that are not present in our model's vocabulary. Write your answer as a single decimal value, rounded to 3 decimal places.

$P(Y = \text{spam}, W = \mathbf{w}) = \boxed{0.012}$ $\quad P(Y = \text{ham}, W = \mathbf{w}) = \boxed{0.027}$

$$\begin{aligned} P(Y = \text{spam}, W = \mathbf{w}) &= P(Y = \text{spam}) \prod_i P(W = w_i | Y = \text{spam}) \\ &= 0.5 \cdot 0.25 \cdot 0.4 \cdot 0.3 \cdot 0.8 \\ &= 0.012 \end{aligned}$$

$$\begin{aligned} P(Y = \text{ham}, W = \mathbf{w}) &= P(Y = \text{ham}) \prod_i P(W = w_i | Y = \text{ham}) \\ &= 0.5 \cdot 0.4 \cdot 0.9 \cdot 0.5 \cdot 0.3 \\ &= 0.027 \end{aligned}$$

(f) To get the conditional class distribution from the joint probabilities in part (e), we normalize $P(Y, W = \mathbf{w})$ by dividing it by some Z, such that $P(Y|W = \mathbf{w}) = \frac{1}{Z} P(Y, W = \mathbf{w})$. Write an expression for Z using any of the following terms: $P(Y = \text{ham}, W = \mathbf{w})$, $P(Y = \text{spam}, W = \mathbf{w})$, $P(Y = \text{ham})$, $P(Y = \text{spam})$, and the integer 1.

$Z = \boxed{P(Y = \text{spam}, W = \mathbf{w}) + P(Y = \text{ham}, W = \mathbf{w})}$

To normalize the joint probability/factor, we sum up all of the joint probabilities over Y.

After training the binary bag-of-words model, you find that the test accuracy is still low.

(g) Instead of treating each word as a feature, you decide to use n-grams of words instead. You then split your labeled dataset into a large training set and a small validation set.

Which of the following is the best way of identifying the optimal value of n for your n-gram model?

- [✓ Train the model on the training data using different values of n; select the n with the highest validation accuracy.]
- ○ Train the model on the training data using different values of n; select the n with the highest training accuracy.
- ○ Select the n that maximizes the number of sequences of n repeated words in the training data.
- ○ Select n to be the average number of characters per word divided by 2.

The value of n is a hyperparameter, since it's a parameter that changes the model itself, as opposed to a parameter within the model that can be learned from data.

We can identify the optimal hyperparameter value by performing hyperparameter tuning, i.e. trying lots of different values of n and picking the value that results in the highest validation accuracy.

Note that we use validation accuracy instead of training accuracy, since we want to know how well the chosen hyperparameter value performs on unseen data. Recall that validation data is labeled data that we intentionally do not use during training, so that we can check our trained model on unseen data and use the labels to see how good the model is.

(h) You apply Laplace smoothing on the bag-of-words data. Select all true statements.

- ☐ Laplace smoothing for bag-of-words always leads to overfitting.
- ☐ Laplace smoothing is applied by subtracting a constant positive value from each word count.
- ☐ Laplace smoothing eliminates the need for a validation set.
- ☐ Laplace smoothing is only useful for large-vocabulary training datasets.
- [✓ None of the above.]

Option 1 is false. There's no guarantee that Laplace smoothing leads to overfitting. In fact, in practice, Laplace smoothing should help us avoid overfitting, by reducing the model's dependence on unseen words.

Option 2 is false. Laplace smoothing involves adding a value to each word count, not subtracting.

Option 3 is false. We still need a validation set to see what value of k works best in Laplace smoothing.

Option 4 is false. In fact, Laplace smoothing is generally more useful for smaller-vocabulary training datasets, where there are lots of words never encountered in the training dataset.