Q7. [17 pts] 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) [3 pts] 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.

(b) [2 pts] 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.

(c) [3 pts] 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.

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) [1 pt] Fill in the following table with integers corresponding to the bag-of-words vector $\mathbf{w}$ for the new email. Ignore punctuation and capitalization.

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

As we are using binary bag-of-words, "team" is only recorded once.

(e) [3 pts] 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.

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

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

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

(f) [2 pts] 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.

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

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) [1 pt] 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.

(h) [2 pts] 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.

Q7. [13 pts] Machine Learning: Hotdog vs. Not Hotdog

Bob is building a model to classify whether a picture contains a Hotdog or not. He uses two binary features: whether the picture has brown color in it and whether there is red color in it. He collects this training set:

Brown Color (W_1)	**Red Color (W_2)**	**Label (y)**
1	0	not hotdog
1	0	not hotdog
1	0	not hotdog
1	1	not hotdog
1	1	hotdog
0	0	hotdog

(a) [5 pts] He first builds a Naive Bayes model. Calculate the following probabilities.

(i) [1 pt] $P(y = \text{hotdog}) =$ $\frac{1}{3}$ $P(y = \text{not hotdog}) =$ $\frac{2}{3}$

(ii) [4 pts] $P(W_1 = 1 \mid y = \text{hotdog}) =$ $\frac{1}{2}$ $P(W_1 = 0 \mid y = \text{hotdog}) =$ $\frac{1}{2}$

$P(W_2 = 1 \mid y = \text{hotdog}) =$ $\frac{1}{2}$ $P(W_2 = 0 \mid y = \text{hotdog}) =$ $\frac{1}{2}$

$P(W_1 = 1 \mid y = \text{not hotdog}) =$ 1 $P(W_1 = 0 \mid y = \text{not hotdog}) =$ 0

$P(W_2 = 1 \mid y = \text{not hotdog}) =$ $\frac{1}{4}$ $P(W_2 = 0 \mid y = \text{not hotdog}) =$ $\frac{3}{4}$

(b) [3 pts] Next, he uses the model to classify three pictures that are from the test set. Fill in the predicted labels in the table.

Test set

Brown Color (W_1)	**Red Color (W_2)**	**Predicted Label ($\hat{y}$)**
1	1	not hotdog
0	1	hotdog
0	0	hotdog

(c) [5 pts] Bob then adds two new examples to his training set, as shown below.

Additional training examples

Brown Color (W_1)	**Red Color (W_2)**	**Label (y)**
0	0	not hotdog
0	1	not hotdog

(i) [3 pts] Now re-classify the test set using the new larger training set (hint: don't forget to update the prior for each class with the new dataset).

Test set

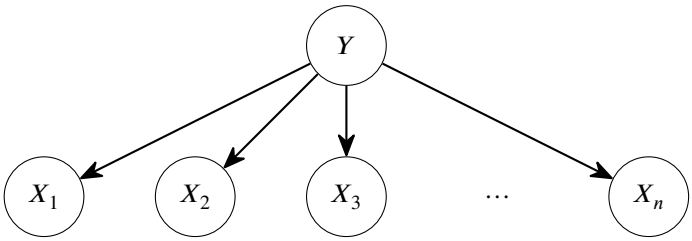
Brown Color (W_1)	**Red Color (W_2)**	**Predicted Label ($\hat{y}$)**
1	1	not hotdog
0	1	not hotdog
0	0	not hotdog

(ii) [2 pts] Did any of the predictions change? Why?

Yes, the last two predictions switched from hotdog to not hotdog. The main reason is because when the label was not hotdog there were no 0's for W_1 in the training set. So, those two test examples originally had 0 probability for the not hotdog prediction and now have much higher probability.

Q8. [13 pts] Inverted Naive Bayes

Consider a standard naive Bayes model with $n > 10$ Boolean features $X_1, \ldots, X_n$ and a Boolean class variable Y. In this question, Boolean variables have values 0 or 1.



(a) [1 pt] How many parameters do we need to learn in this model (not counting parameters that can be derived by the sum-to-1 rule)?

Example of sum-to-1 rule: Given $P(Y = 0)$, we can compute $P(Y = 1)$ since we know that these two values sum to 1. Therefore, $P(Y = 0)$ and $P(Y = 1)$ only count as one parameter we need to learn.

Your answer should be an expression, possibly in terms of n.

$2n + 1$

Under the Y node, there is a conditional probability table $P(Y)$ with two rows: $P(Y = 0)$ and $P(Y = 1)$. We must learn one of these as parameters; as soon as we learn one, the other one can be derived through the sum-to-one rule.

Under each X_i node, there is a conditional probability table $P(X_i|Y)$ with four rows. $P(X_i = 0|Y = 0)$ and $P(X_i = 1|Y = 0)$ sum to 1, so we have to learn one parameter for these two values. Also, $P(X_i = 0|Y = 1)$ and $P(X_i = 1|Y = 0)$ sum to 1, so we have another parameter to learn for these two values. In total, there are 2 parameters to learn for each X_i node.

There is 1 parameter to learn from the one Y node. There are 2 parameters to learn for each of the X_i nodes, and there are n of those, for a total of $2n$ parameters from the X_i nodes. In total, that's $2n + 1$ parameters.

(b) [2 pts] We observe one training example with features $x_1, \ldots, x_n$ and class y. Fill in the blanks to derive the likelihood of this training example.

$$L = P(x_1, \ldots, x_n, y) = P(Y = 1)^{\textbf{(i)}}\ P(Y = 0)^{\textbf{(ii)}} \prod_{i=1}^{n} P(X_i = 1|y)^{\textbf{(iii)}}\ P(X_i = 0|y)^{\textbf{(iv)}}$$

(i)	○ x	○ $1 - x$	● y	○ $1 - y$
(ii)	○ x	○ $1 - x$	○ y	● $1 - y$
(iii)	● x	○ $1 - x$	○ y	○ $1 - y$
(iv)	○ x	● $1 - x$	○ y	○ $1 - y$

Clarification during exam: The x in the answer choices should say x_i.

The probability requested here $P(x_1, \ldots, x_n, y)$ is an entry in the joint distribution. To obtain an entry in the joint distribution, we just need to multiply the corresponding entry from each of the conditional probability tables together.

First, consider the CPT under node Y, which is $P(Y)$. If $y = 0$, the CPT entry we need is $P(Y = 0)$. If $y = 1$, the CPT entry we need is $P(Y = 1)$. To write this if/else condition into our equation, we can write $P(Y = 1)^y P(Y = 0)^{1-y}$.

Note that if $y = 0$, then the first term is equal to 1 (raised to 0th power), and the second term is raised to the $1 - 0 = 1$st power, for $P(Y = 0)$, as desired. Also, if $y = 1$, then the first term is raised to the 1st power, and the second term is equal to 1 (raised to the 0th power), for $P(Y = 1)$, as desired.

Next, consider the CPTs under each X_i node. Again, if $x_i = 0$, we need the CPT entry $P(X_i = 0|y)$, and if $x_i = 1$, we need the CPT entry $P(X_i = 1|y)$. Using the same trick as we did for $P(y)$, we can write $P(X_i = 1|y)^{x_i} P(X_i = 0|y)^{1-x_i}$.

For each of the n CPTs corresponding to X_i nodes, we need to multiply one entry per CPT, which is the summation operator in the existing expression.

(c) [1 pt] Suppose that this training example was missing a feature value, so we only observed $x_2, \ldots, x_n$ and class y.

How is the likelihood with a missing feature $L' = P(x_2, \ldots, x_n, y)$, related to the original likelihood L?

- ○ L' is equal to L, but with any terms involving $P(X_1|y)$ dropped.
- ○ L' is equal to L, with an extra term related to the prior probabilities $P(X_1)$, and any terms involving $P(X_1|y)$ dropped.
- ● The correct expression for L' cannot be determined from L.

$L' = P(x_2, \ldots, x_n, y) = \sum_{x_1} P(x_1, x_2, \ldots, x_n, y)$.

In words: L' is an entry in the marginal distribution where we summed x_1 out of the joint distribution. In other words, we took the joint distribution, collected all rows with values $x_2, \ldots, x_n, y$ (there will be one row of this per value of x_1, for two rows in total), and summed up all their values.

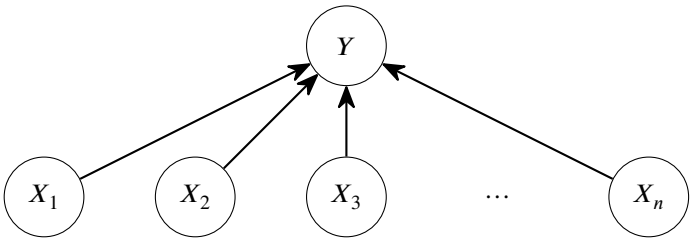
L, with a fixed value of x_1, corresponds to just one of the many rows that we summed together to get L'. With just L, we cannot deduce what the other values to sum together are, so we cannot derive L' from L.

Another way to think about this is to actually write out the summation over x_1, which is possible here since there's only two possible values of x_1:

$L' = P(X_1 = 0, x_2, \ldots, x_n, y) + P(X_1 = 1, x_2, \ldots, x_n, y)$

L is going to correspond to one of these two values (depending on if x_1 is 0 or 1). Given just L, we have no way to work out what the other value in the summation is to derive L'.

Suppose we "invert" the naive Bayes model so that the arrows point from the feature variables to the class variable:



(d) [1 pt] How many parameters do we need to learn in this model (not counting parameters that can be derived by the sum-to-1 rule)?

Your answer should be an expression, possibly in terms of n.

$n + 2^n$

Following simliar logic as part (a) here:

Under each X_i node, we have $P(X_i)$, a CPT with two rows. The two rows have to sum to 1, so each X_i node has one parameter we need to learn. There are n of these X_i nodes, so that's n parameters in total from all the X_i nodes.

$P(Y|X_1, X_2, \ldots, X_n)$ is a CPT with 2^{n+1} rows. For a fixed set of $x_1, \ldots, x_n$, the two rows $P(Y = 0|x_1, \ldots, x_n)$ and $P(Y = 1|x_1, \ldots, x_n)$ must sum to 1, so we only have to learn half of the rows in the table as parameters, and the other half can be derived from the sum-to-1 rule. In total, that's 2^n parameters (corresponding to half of the 2^{n+1} rows).

There are n parameters from the X_i nodes and 2^n parameters from the Y node, for a total of $n + 2^n$ parameters.

For the rest of this question, Boolean variables are represented as positive or negative (e.g. $+y$ and $-y$).

Suppose you have all the parameters from the original naive Bayes model. Using any relevant parameters from the original model, derive the following parameters in the inverted naive Bayes' model, so that the two models represent the same joint distribution.

Your answers should be expressions, possibly in terms of $P(+y)$, $P(-y)$, $P(+x_i|+y)$, $P(-x_i|+y)$, $P(+x_i|-y)$, and $P(-x_i|-y)$, for $1 \leq i \leq n$. Hint: You can also use summations $\sum$ and products $\prod$.

(e) [2 pts] $P(+x_i) =$ $\sum_y P(+x_i|y)P(y)$

One way to solve this is to look at the CPT values in the original Bayes' model (also listed above the question). From this list, note that the closest value to $P(x_i)$ is the CPT under the x_i node, which is $P(x_i|Y)$. Then, to get a marginal distribution over x_i, you can sum over the variable you don't care about (y).

$P(+x_i) = \sum_y P(+x_i|y)P(y)$

Alternatively, if you wrote out the summation over y explicitly, you get:

$P(+x_i) = P(+x_i| + y)P(+y) + P(+x_i| - y)P(-y)$

(f) [2 pts] $P(+y| + x_1, +x_2, \ldots, +x_n) \propto$ $P(+y) \prod_{i=1}^{n} P(+x_i| + y)$

One solution is to use Bayes' rule, since we need an expression with y on the left and x_i on the right of the conditioning bar, but our the listed expressions all have x_i on the left and y on the right.

$P(+y| + x_1, \ldots, +x_n) = \frac{P(+y)P(+x_1,\ldots,+x_n|+y)}{P(+x_1,\ldots,x_n)}$

Now, note that the denominator is a constant even if we vary y, and the question only asks for a proportional value ($\propto$, not $=$), so we can safely ignore the denominator. (Recall the normalization trick we showed in the first probability lecture for why this works.)

Looking at the numerator, $P(+x_1, \ldots, +x_n | + y)$ can be decomposed into a product of individual terms $P(+x_i | + y)$, for the same reason that this decomposition works in the standard Bayes' net. This leaves us with:

$P(+y) \prod_{i=1}^{n} P(+x_i | + y)$

Another quick solution is to note that the desired expression $P(+y | +x_1, \ldots, +x_n)$ is exactly the expression for classification in the original model. So all we have to do is write the classification for expression in the original model.

(g) [3 pts] Select all true statements.

- [x] For any set of parameters in the original naive Bayes model, there is some set of parameters in the inverted naive Bayes model that captures the same joint distribution.
- [ ] For any set of parameters in the inverted naive Bayes model, there is some set of parameters in the original naive Bayes model that captures the same joint distribution.
- [x] The inverted naive Bayes model will typically require far more training data than the original naive Bayes model to achieve the same level of test accuracy.
- ○ None of the above

For options (A) and (B), the quickest intuitive (but definitely not rigorous) answer is to note that the inverted Bayes' net has far more parameters and is probably going to have more expressive power than the normal Bayes' net. Therefore, all distributions representable in the simpler original model should still be representable in the more complex inverted model. However, not all distributions representable in the more complex inverted model can be represented in the simpler original model.

A more rigorous proof for (A) is to note that in the previous two subparts, we were able to derive every parameter in the inverted Bayes' net from the values in the original Bayes' net. Therefore, given any values in the original Bayes' net, we can derive values in the inverted Bayes' net that represent the same joint distribution.

(C): True. Intuitively, the inverted model has far more parameters to learn, so we need more training data to learn the parameters well.

More rigorously, in the original Bayes' net, we just have tables $P(Y)$ and $P(X_i|Y)$. To learn the $P(Y)$ table accurately, we just need to count the $+y$ data points $-y$ data points, and a large enough dataset should have a nontrivial number of data points from each class. To learn $P(X_i|Y)$ accurately, we just need lots of data points with each of these four configurations: $(+x, +y), (-x, +y), (+x, -y), (-x, -y)$. Again, with a large enough data set, there should be a nontrivial number of data points with each of these four possible configurations, which would allow us to take counts and estimate parameters well.

However, in the inverted Bayes' net, we have to learn the values in the table $P(Y|X_1, \ldots, X_n)$. Consider just one pair of rows: $P(+y|x_1, \ldots, x_n)$ and $P(-y|x_1, \ldots, x_n)$, for some fixed $x_1, \ldots, x_n$. To learn these two values accurately, we would need lots of data points with exactly $x_1, \ldots, x_n$ so that we can count how many have $+y$ and how many have $-y$. It's going to be pretty unusual to find many data points with exactly this set of feature values; furthermore, even if we found such data points, they would be useless for learning any other parameter in the table.

(h) [1 pt] What learning behavior should we expect to see with this inverted naive Bayes' model?

- ○ High training accuracy, high test accuracy
- ● High training accuracy, low test accuracy
- ○ Low training accuracy, high test accuracy
- ○ Low training accuracy, low test accuracy

With a sizable number of features, what will probably happen is that no two inputs will have the exact same feature values, or very few inputs will have the same exact feature values. Consider the case where no two inputs have the exact same feature values. For some given training data point $x_1, \ldots, x_n$, $P(Y|x_1, \ldots, x_n)$ derived from counts will equal 1 for this data point's true class, and 0 for the other class. This will give us high training accuracy; when we try to classify that data point $x_1, \ldots, x_n$, the CPT will directly tell us its true class (which it learned from training).

However, this model will probably perform poorly on test data that it hasn't seen before. For some unseen training data point $x_1, \ldots, x_n$, $P(Y|x_1, \ldots, x_n)$ will be 0 for both values of Y, which results in lots of "ties" or undefined classifications.

In other words, this model is overfitting: the CPT under Y is essentially memorizing all the true classes of the training data points, and is not generalizing well to data points it hasn't seen before. Smoothing helps somewhat.