1 Question 1: Course Gossip (9 points)

In this question, you will train classifiers to predict whether sentences are about CS 188 or CS 186 (databases) using a bag-of-words Naive Bayes classifier. Each sentence is labeled with the class to which it pertains.

<table>
<thead>
<tr>
<th>Training set</th>
<th>Held-out set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>(188) agents need good models.</td>
<td>(188) agents need memory.</td>
<td>(186) data have data models.</td>
</tr>
<tr>
<td>(188) agents need data.</td>
<td>(186) DBs have data.</td>
<td></td>
</tr>
<tr>
<td>(186) buffers need memory.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(186) DBs need data models.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

a) Write down all of the maximum likelihood (relative frequency) parameters for a bag-of-words naive Bayes classifier trained on the training set above. Let \( Y \) be the class for a sentence, and \( W \) be a word. You may omit any parameters equal to 0. Ignore punctuation. Note: Bag-of-words classifiers assume that the words at every sentence position are identically distributed. Repeated words affect both the likelihood of a word during estimation and sentence probabilities during inference.

| \( W \) | \( P(W | Y = 188) \) | \( W \) | \( P(W | Y = 186) \) |
|---------|----------------|---------|----------------|
| agents  | \( \frac{7}{8} \) | need    | \( \frac{7}{8} \) |
| good    | \( \frac{7}{8} \) | buffers | \( \frac{7}{8} \) |
| models  | \( \frac{7}{8} \) | memory  | \( \frac{7}{8} \) |
| classifiers | \( \frac{7}{8} \) | DBs     | \( \frac{7}{8} \) |
| data    | \( \frac{7}{8} \) | data    | \( \frac{7}{8} \) |
| models  | \( \frac{7}{8} \) | data    | \( \frac{7}{8} \) |

b) According to your classifier, what is the probability that the first held-out sentence “agents need memory” is about 188?

Since \( P(W = \text{memory} | Y = 188) = 0 \), the joint probability

\[
P(Y = 188, W_1 = \text{agents}, W_2 = \text{need}, W_3 = \text{memory}) = 0
\]

Likewise, since \( P(W = \text{agents} | Y = 186) = 0 \), the joint probability

\[
P(Y = 186, W_1 = \text{agents}, W_2 = \text{need}, W_3 = \text{memory}) = 0
\]

Adding these together, we find that according to our model, \( P(W_1 = \text{agents}, W_2 = \text{need}, W_3 = \text{memory}) = 0 \). Therefore, the posterior probability \( P(Y = 188 | W_1 = \text{agents}, W_2 = \text{need}, W_3 = \text{memory}) \) is undefined: \( \frac{0}{0} \). The answer 0 will also be accepted.
c) Using Laplace (i.e., add one) smoothing for all of your parameters, what is the probability of seeing the test sentence “data have data models”: \( P(W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) \)? Assume that the only words you ever expect to see are those in your training and held-out sets. *Hint: sum over \( Y \), using the new estimates after smoothing.*

\( W \) can take 9 different values, so we use the following smoothed parameters to compute the quantity desired:

\[
\begin{array}{c|c|c|c}
Y & P(Y) & P(W | Y = 188) & P(W | Y = 186) \\
188 & \frac{5}{8} & \frac{1}{5} & \frac{1}{5} \\
186 & \frac{3}{8} & \frac{1}{5} & \frac{1}{5} \\
\end{array}
\]

\[
P(W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) \\
= P(Y = 188, \text{data}, \text{have}, \text{data}, \text{models}) + P(Y = 186, \text{data}, \text{have}, \text{data}, \text{models}) \\
= P(188)P(\text{data}|188)^2P(\text{have}|188)P(\text{models}|188) + P(186)P(\text{data}|186)^2P(\text{have}|186)P(\text{models}|186) \\
= \frac{1}{214} \cdot \frac{1}{214} + \frac{1}{214} \cdot \frac{1}{214} = \frac{1}{213}
\]

*Note: Because the wording of the question was confusing, we will also accept answers based on the following (incorrect) smoothed distribution, which uses word counts from the validation set in addition to the training set to estimate probabilities in the model. In general, the validation set should never be used to compute counts. It exists to test performance of a classifier trained on the training data.*

\[
\begin{array}{c|c|c|c}
Y & P(Y) & P(W | Y = 188) & P(W | Y = 186) \\
188 & \frac{5}{8} & \frac{2}{19} & \frac{1}{19} \\
186 & \frac{3}{8} & \frac{2}{19} & \frac{1}{19} \\
\end{array}
\]

\[
P(W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) \\
= P(Y = 188, \text{data}, \text{have}, \text{data}, \text{models}) + P(Y = 186, \text{data}, \text{have}, \text{data}, \text{models}) \\
= P(188)P(\text{data}|188)^2P(\text{have}|188)P(\text{models}|188) + P(186)P(\text{data}|186)^2P(\text{have}|186)P(\text{models}|186) \\
= \frac{4}{19^4} + \frac{18}{19^4} = \frac{23}{19^4}
\]
d) Using Laplace smoothing, what is the probability according to your classifier that the test sentence “data have data models” is about 186?
   From the previous question, we have:
   
   \[
   P(Y = 186, W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) = \frac{1}{214}
   \]
   
   \[
   P(W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) = \frac{1}{215}
   \]

   Hence, the conditional probability
   
   \[
   P(Y = 186 | W_1 = \text{data}, W_2 = \text{have}, W_3 = \text{data}, W_4 = \text{models}) = \frac{1}{2}
   \]

   which will only classify the example correctly if we happen to break ties correctly (not a method we want to rely on).

   Using the alternate values for 1(c), we instead get \( \frac{9}{11} \), which classifies the sentence correctly.

---

e) Suggest an additional feature that would allow the classifier to correctly conclude that “data have data models” is about 186 when trained on this training set.

The bigram feature “data models” would suffice, as it only appears in data labeled 186. Several other features are also acceptable, like a feature for the absence of the word “agents”. In general, any feature that favors 186 over 188 in the training data that is also relevant for the test datum we are trying to satisfy would work.
2 Question 2: Red Light, Green Light (7 points)

You meet the chief administrative officer for intersection oversight for the California department of transportation. Her job is to report whether the stoplights at intersections are working correctly. You remark, “I bet I could automate your job.” She scoffs in disbelief, but humors you by showing you some data.

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>red (-1)</td>
<td>red (-1)</td>
<td>broken (+1)</td>
</tr>
<tr>
<td>red (-1)</td>
<td>green (+1)</td>
<td>working (-1)</td>
</tr>
<tr>
<td>green (+1)</td>
<td>red (-1)</td>
<td>working (-1)</td>
</tr>
<tr>
<td>green (+1)</td>
<td>green (+1)</td>
<td>broken (+1)</td>
</tr>
</tbody>
</table>

**Discussion:** Binary classification is a special case of multi-class classification, in which the labels are $-1$ and $+1$. It is always the case that $w_{-1} = -1 \cdot w_{+1}$, and so we only track and update $w_{+1}$ by convention. The update rules for perceptron and MIRA are the same as for multi-class.

**a)** The data above have been annotated with feature and output values. $Y$ is the label variable to predict. Circle all of the following feature sets make the training data linearly separable.

(i) $X_1, X_2$  
(ii) $X_1$ only  
(iii) $\min(X_1, X_2), X_1, X_2$  
(iv) $\min(X_1, X_2), X_2$  
(v) $|X_1 + X_2|$

**Answer:** (iii) and (v).

- (i): These data are not separable with the features provided. Plotting them on a plane with $X_1$ and $X_2$ as axes simply demonstrates that no line can be drawn that separates the broken data from the working data.
- (ii): Both the first and second examples have the same feature $X_1$, but different labels. Therefore, no classifier will be able to correctly label them both.
- (iii): The linear decision rule $y = -X_1 - X_2 + 2 \cdot \min(X_1, X_2) + 1$ gives correct labels to each datum.
- (iv): Both the first and third examples have the same values for the pair of features $[\min(X_1, X_2), X_2]$, but different labels. Therefore, no classifier will be able to correctly label them both.
- (v): The linear decision rule $y = |X_1 + X_2| - 1$ gives correct labels to each datum.

**b)** Using just $X_1$ and $X_2$ as features, fill in the resulting weight vectors after two-class perceptron updates (one vector of weights) for the first and second data points sequentially, starting with the initial vector below.

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Initial Weights</strong></td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td><strong>Weights after observing</strong> ($X_1 = -1, X_2 = -1, Y = 1$)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Weights after observing</strong> ($X_1 = -1, X_2 = 1, Y = -1$)</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Explanation:** The first observation is classified as $-1$: $2 \cdot -1 + 1 \cdot -1 = -3 < 0$. Since the correct label was $Y = +1$, the datum is added to the initial weight vector. The second datum is classified as $-1$ using the new weights: $1 \cdot -1 + 0 \cdot 1 = -1 < 0$. Since this label is correct, the weights are unchanged.
c) Using just \(X_1\) and \(X_2\) as features, fill in the resulting weight vectors after two-class MIRA updates for the first and second data points sequentially, starting with the initial vector below. Assume the maximum update size \(C\) is 5.

<table>
<thead>
<tr>
<th></th>
<th>(X_1)</th>
<th>(X_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Weights</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Weights after observing ((X_1 = -1, X_2 = -1, Y = 1))</td>
<td>(-\frac{2}{3})</td>
<td>(-\frac{1}{3})</td>
</tr>
<tr>
<td>Weights after observing ((X_1 = -1, X_2 = 1, Y = -1))</td>
<td>0</td>
<td>(-\frac{1}{2})</td>
</tr>
</tbody>
</table>

**Explanation:** The first datum is classified as \(-1\): \(1 \cdot -1 + 2 \cdot -1 = -3 < 0\). We must first compute \(\tau\), then make the update to the weights. Below, vectors are represented as pairs \([X_1, X_2]\). All products are dot products: \([X_1, X_2] \cdot [X'_1, X'_2] = X_1 \cdot X'_1 + X_2 \cdot X'_2\). Because the positive-label weight vector is \([1, 2]\), the negative label weight vector is \([-1, -2]\).

\[
\tau = \min \left\{ 5, \frac{([-1, -2] - [1, 2]) \cdot [-1, -1] + 1}{2 \cdot [-1, -1] \cdot [-1, -1]} \right\}
\]

\[
= \min \left\{ 5, \frac{[-2, -4] \cdot [-1, -1] + 1}{2 \cdot [-1, -1] \cdot [-1, -1]} \right\}
\]

\[
= \min \left\{ 5, \frac{7}{4} \right\} = \frac{7}{4}
\]

We can check our answer in the following manner. After making a MIRA update with \(\tau < C\), we should now correctly classify the example we just trained on by a margin of 1. That is,

\[
\begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot [-1, -1] = \begin{bmatrix} 3 \\ 4 \end{bmatrix} \cdot [-1, -1] + 1
\]

We add \(\tau [-1, -1]\) to \(w\) because the correct label is +1. The computation of \(\tau\) above indicates that there is a general form for computing \(\tau\) in binary classification. Let \(y^*\) be the correct label (either \(-1\) or \(+1\)), \(w\) be the weight vector for the classifier, and \(f\) be the feature vector for the datum. Then, we update using

\[
\tau = \min \left\{ C, \frac{-2 \cdot y^* \cdot w \cdot f + 1}{2 \cdot f \cdot f} \right\}
\]

Using the new weight vector, we classify the second datum as \(+1\): \(-\frac{3}{4} \cdot -1 + \frac{1}{4} \cdot 1 > 0\). Therefore, we update again, this time choosing

\[
\tau = \min \left\{ 5, \frac{-2 \cdot [-1, -1] \cdot [-3, 1] + 1}{2 \cdot [-1, -1] \cdot [-1, -1]} \right\} = \min \left\{ 5, \frac{2 \cdot 1 + 1}{2 \cdot 2} \right\} = 3
\]

This time, we subtract \(\tau [-1, 1]\) from \(w\). Again, if we were to classify \([X_1 = -1, X_2 = 1]\) with these new weights, we would be correct by a margin of 1.
d) Is it possible in any classification problem for a perceptron classifier to reach perfect training set accuracy in fewer updates than a MIRA classifier, if they both start with the same initial weights and examine the training data in the same order? Briefly justify your answer.

Yes. MIRA makes conservative updates to control the overcorrecting behavior sometimes exhibited by perceptron updates, but an overcorrecting update can fortuitously reach a separating solution faster than a conservative one. Consider the following dataset:

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-\frac{1}{2}$</td>
<td>$-\frac{1}{2}$</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

If we start weights at 0 and break ties by always using the positive class, then both MIRA and Perceptron will update on the first example to weights $[-\frac{1}{2}, -\frac{1}{2}]$. On the second example, they will differ. The perceptron will update to $[-\frac{1}{2}, \frac{3}{2}]$. This new weight vector classifies all three examples correctly.

On the other hand, MIRA will make a smaller update:

$$\tau = -2 \cdot \left[-\frac{1}{2} \cdot \frac{1}{2} \cdot [0, 2] + 1\right] = \frac{3}{8}$$

Thus, MIRA will update to $[-\frac{1}{2}, \frac{1}{2}]$, which is enough of a change to classify the second example correctly by a margin, but not enough to also classify the third example correctly.

A similar effect can be observed when using a small value for $C$, which forces MIRA to make very small updates.
3 Question 3: One-Sentence Answers (4 points)

a) What is one advantage of naive Bayes over MIRA?
   Naive Bayes is a probabilistic model, while MIRA is a linear classifier. The parameters of a naive Bayes model — the prior and conditional probability tables — have a clear interpretation as statistical properties of the training set. Naive Bayes also gives posterior probabilities for each class, which have a probabilistic interpretation, instead of just activations. Finally, naive Bayes requires only a single pass through the training set (assuming fixed smoothing hyperparameters), while MIRA requires several passes.

b) What is one advantage of MIRA over Naive Bayes?
   MIRA does not make any independence assumptions about the features of the dataset, and so better accommodates features that are in fact dependent conditioned on the label. MIRA is also a mistake-driven discriminative training procedure, rather than a generative model of the training data. Discriminative methods perform better on many real-world classification problems. In the limit of training, MIRA is guaranteed to find a separating solution for any linearly separable training set, while naive Bayes models come with no such guarantee.

c) What is one advantage of a nearest-neighbor classifier over Naive Bayes?
   Nearest-neighbor classifiers can use any measure of similarity between examples, rather than the feature-based posterior probability used by naive Bayes. General similarity functions, which can be computed with any snippet of code we choose to write, can offer greater expressiveness than features, and with clever design can improve performance. Additionally, nearest-neighbor classifiers are non-parametric: they express increasingly rich decision functions when trained on more data, while parametric classifiers like naive Bayes have the same number of parameters (and therefore the same expressiveness) regardless of the size of the training set.

d) Invent a method of combining a Naive Bayes classifier with a MIRA classifier and describe it.
   Two common approaches are to let the two classifiers vote on the label. These votes can be weighted, perhaps by the confidence of the classifiers, or perhaps using a third classifier that takes the output of the naive Bayes classifier and the MIRA classifier as two input features. Similarly, the output label of the naive Bayes classifier can be used as an additional feature for the MIRA classifier, or vis versa.