Homework 3

CMSC 478 – Machine Learning

Due Oct 4, 2024

Item	**Summary**
Assigned	Sep 26, 2024
Due	Oct 4, 11:59 PM
Topic	Naive Bayes
Points	100

You are to complete this assignment on your own: that is, the writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or blackboard discussions). *Courtesy:* This assignment is adapted from Professor KMA Solaiman.

What To Turn In You must turn in two items on Blackboard:

- A writeup in PDF format that answers the questions.
- The filled-up notebook file (.ipynb) with your code.
- Provide any environment files (if you used external ones), and execution instructions necessary to replicate your output.

Problem 1: Implementing Naive Bayes

In this question you will implement a Naive Bayes classifier for a text classification problem. You will be given a collection of text articles, each coming from either the serious European magazine *The Economist*, or from the not-so-serious American magazine *The Onion*. The goal is to learn a classifier that can distinguish between articles from each magazine.

We have pre-processed the articles so that they are easier to use in your experiments. We extracted the set of all words that occur in any of the articles. This set is called the *vocabulary* and we let V be the number of words in the vocabulary. For each article, we produced a feature vector $X = \langle X_1, \ldots, X_V \rangle$, where

$$X_i = \begin{cases} 1, & \text{if the } i^{\text{th}} \text{ word appears in the article} \\ 0, & \text{otherwise} \end{cases}$$

and each article is also accompanied by a class label y_i, where

$$y_i = \begin{cases} 1, & \text{for The Economist} \\ 2, & \text{for The Onion} \end{cases}$$

When we apply the Naive Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data is drawn iid from a joint probability distribution over the possible feature vectors X and the corresponding class labels Y; second, we assume for each pair of features X_i and X_j with $i \neq j$

that X_i is conditionally independent of X_j given the class label Y (this is the Naive Bayes assumption). Under these assumptions, a natural classification rule is as follows: Given a new input X, predict the most probable class label $\hat{Y}$ given X. Formally,

$$\hat{Y} = \operatorname*{argmax}_y P(Y = y|X).$$

Using Bayes Rule and the Naive Bayes assumption, we can rewrite this classification rule as follows:

$$\begin{aligned}
\hat{Y} &= \operatorname*{argmax}_y \frac{P(X|Y = y)P(Y = y)}{P(X)} && \text{(Bayes Rule)} \\
&= \operatorname*{argmax}_y P(X|Y = y)P(Y = y) && \text{(Denominator does not depend on } y\text{)} \\
&= \operatorname*{argmax}_y P(X_1, \ldots, X_V|Y = y)P(Y = y) \\
&= \operatorname*{argmax}_y \left(\prod_{w=1}^{V} P(X_w|Y = y) \right) P(Y = y) && \text{(Conditional independence).}
\end{aligned}$$

Of course, since we don't know the true joint distribution over feature vectors X and class labels Y, we need to estimate the probabilities $P(X|Y = y)$ and $P(Y = y)$ from the training data. For each word index $w \in \{1, \ldots, V\}$ and class label $y \in \{1, 2\}$, the distribution of X_w given $Y = y$ is a Bernoulli distribution with parameter θ_{yw}. In other words, there is some unknown number θ_{yw} such that

$$P(X_w = 1|Y = y) = \theta_{yw}$$
$$P(X_w = 0|Y = y) = 1 - \theta_{yw}.$$

We believe that there is a non-zero (but maybe very small) probability that any word in the vocabulary can appear in an article from either The Onion or The Economist. To make sure that our estimated probabilities are always non-zero, we will impose a Beta(2,1) prior on θ_{yw} and compute the MAP estimate from the training data.

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution with parameter ρ. In other words, there is some unknown number ρ such that

$$P(Y = 1) = \rho$$

$$P(Y = 2) = 1 - \rho.$$

In this case, since we have many examples of articles from both The Economist and The Onion, there is no risk of having zero-probability estimates, so we will instead use the MLE for the distribution of Y.

Programming Instructions

Each of these questions ask you to implement one function related to the Naive Bayes classifier.

You will submit the edited Python notebook online through Blackboard. The file `HW4Data.mat` contains the data that you will use in this problem. You can load it using the *data loading cells* provided in the notebook. After loading the data, you will see that there are 7 variables: `Vocabulary`, `XTrain`, `yTrain`, `XTest`, `yTest`, `XTrainSmall`, and `yTrainSmall`.

- `Vocabulary` is a $V \times 1$ dimensional cell array that that contains every word appearing in the documents. When we refer to the j^{th} word, we mean `Vocabulary(j,1)`.
- `XTrain` is a $n \times V$ dimensional matrix describing the n documents used for training your Naive Bayes classifier. The entry `XTrain(i,j)` is 1 if word j appears in the i^{th} training document and 0 otherwise.
- `yTrain` is a $n \times 1$ dimensional matrix containing the class labels for the training documents. `yTrain(i,1)` is 1 if the i^{th} document belongs to The Economist and 2 if it belongs to The Onion.

- XTest and yTest are the same as XTrain and yTrain, except instead of having n rows, they have m rows. This is the data you will test your classifier on and it should not be used for training.
- Finally, XTrainSmall and yTrainSmall are subsets of XTrain and yTrain which are used in the final question.

Logspace Arithmetic

When working with very large or very small numbers (such as probabilities), it is useful to work in *logspace* to avoid numerical precision issues. In logspace, we keep track of the logs of numbers, instead of the numbers themselves. For example, if $p(x)$ and $p(y)$ are probability values, instead of storing $p(x)$ and $p(y)$ and computing $p(x) * p(y)$, we work in log space by storing $\log(p(x))$, $\log(p(y))$, and we can compute the log of the product, $\log(p(x) * p(y))$ by taking the sum: $\log(p(x) * p(y)) = log(p(x)) + log(p(y))$.

Questions

1. Complete the function logProd(x). The function logProd(x) takes as input a vector of numbers in logspace (i.e., $x_i = \log p_i$) and returns the product of those numbers in logspace—i.e., logProd(x) $= \log(\prod_i p_i)$. **[1 point]**
2. Complete the function [D] = NB_XGivenY(XTrain, yTrain). The output D is a $2 \times V$ matrix, where for any word index $w \in \{1, \ldots, V\}$ and class index $y \in \{1, 2\}$, the entry D(y,w) is the MAP estimate of θ_{yw} with a $Beta(2, 1)$ prior distribution. **[4 points]**
3. Complete the function [p] = NB_YPrior(yTrain). The output p is the MLE estimate for ρ. **[4 points]**
4. Complete the function [yHat] = NB_Classify(D, p, X). The input X is an $m \times V$ matrix containing m feature vectors (stored as its rows). The output yHat is a $m \times 1$ vector of predicted class labels, where yHat(i) is the predicted label for the i^{th} row of X. [Hint: In this function, you will want to use the logProd function to avoid numerical problems.] **[8 points]**
5. Complete the function [error] = ClassificationError(yHat, yTruth), which takes two vectors of equal length and returns the proportion of entries that they disagree on. **[1 point]**
6. Train your classifier on the data contained in XTrain and yTrain by running

   ```
   D = NB_XGivenY(XTrain, yTrain);
   p = NB_YPrior(yTrain);
   ```

   Use the learned classifier to predict the labels for the article feature vectors in XTrain and XTest by running

   ```
   yHatTrain = NB_Classify(D, p, XTrain);
   yHatTest = NB_Classify(D, p, XTest);
   ```

   Use the function ClassificationError to measure and report the training and testing error by running

   ```
   trainError = ClassificationError(yHatTrain, yTrain);
   testError = ClassificationError(yHatTest, yTest);
   ```

   How do the train and test errors compare? Explain the reasoning behind any significant differences. **[4 points]**
7. Repeat the steps from question 6, but this time use the smaller training set XTrainSmall and yTrainSmall. Explain any difference between the train and test error in this question and in question 6. [Hint: When we have less training data, does the prior have more or less impact on our classifier?] **[4 points]**

8. Finally, we will try to interpret the learned parameters. Train your classifier on the data contained in `XTrain` and `yTrain`. For each class label $y \in \{1, 2\}$, list the five words that the model says are most likely to occur in a document from class y. Also for each class label $y \in \{1, 2\}$, list the five words w that maximize the following quantity:

$$\frac{P(X_w = 1|Y = y)}{P(X_w = 1|Y \neq y)}.$$

Which list of words describes the two classes better? Briefly explain your reasoning. (Note that some of the words may look a little strange because we have run them through a stemming algorithm that tries to make words with common roots look the same. For example, "stemming" and "stemmed" would both become "stem".) **[4 points]**