10-401 Machine Learning: Homework 1

Due 12 p.m. Thursday, February 11, 2016

100 points

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date and zero credit after that. You have four late days to use throughout the entire semester.
- **Collaboration policy:** You may discuss the questions with other students. However, you are responsible for writing your own answers and code. Do not search for answers to homework questions online. If you are not sure if you can use a particular reference, ask us.
- **Online submission:** You must submit your solutions online on Autolab. On the Homework 1 Autolab page, you can download the template, which is a tar archive containing one Octave source file for each of the programming questions. Fill in each of the Octave source files. When you are ready to submit, create a new tar archive of the top-level directory and submit your archived solutions online by clicking the "Submit File" button. You should submit a single tar archive identical to the template, except with each of the Octave source files filled in.

  DO NOT change the name of any of the files or folders in the submission template. In other words, your submitted files should have exactly the same names as those in the submission template. Do not modify the directory structure.

Problem 1: Multiple Choice and Short Answer Questions

[**48 Points**] Please visit http://bit.ly/HW1_qna to answer 12 multiple choice and short answer questions. Each question is worth 4 points when answered correctly at the first attempt and 1/2 credit (2 points) when eventually answered correctly.

Problem 2: Implementing Naive Bayes

[**52 Points**] 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 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 is equal to 1 if the i^{th} word appears in the article and 0 otherwise. Each article is also accompanied by a class label of either 0 for The Economist or 1 for The Onion.

When we apply the Naive Bayes classification algorithm, we make two assumptions about the data: first, we assume that our data are 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: For 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:

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

The advantage of the Naive Bayes assumption is that it allows us to represent the distribution $P(X|Y = y)$ using many fewer parameters than would otherwise be possible. Specifically, since all the random variables are binary, we only need one parameter to represent the distribution of X_w given Y for each $w \in \{1, \ldots, V\}$ and $y \in \{0, 1\}$. This gives a total of $2V$ parameters. On the other hand, without the Naive Bayes assumption, it is not possible to factor the probability as above, and therefore we need one parameter for all but one of the 2^V possible feature vectors X and each class label $y \in \{0, 1\}$. This gives a total of $2(2^V - 1)$ parameters. The vocabulary for our data has $V \approx 26,000$ words. Under the Naive Bayes assumption, we require on the order of $52,000$ parameters, while without it we need more than 10^{7000}!

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 $y \in \{0, 1\}$, 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

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

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. A closed-form formula for the MAP estimate of θ_{yw}, here denoted by D_{yw}, is given by

$$D_{yw} = \frac{\#\{X_w = 1, Y = y\} + 1}{\#\{Y = y\} + 1}.$$

[**Note:** Here we present a prior on the conditional distribution $P(X|Y)$. Lecture 4, in the discussion of MAP estimates and adding "virtual" data points, gave priors on the marginal distribution $P(Y)$ and the joint distribution $P(X, Y)$.]

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

$$P(Y = 1) = \rho \quad \text{and} \quad P(Y = 0) = 1 - \rho.$$

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.

Programming Instructions

Parts (a) through (e) of this question each ask you to implement one function related to the Naive Bayes classifier. You will submit your Octave code online through the CMU Autolab system. For further information on Octave and Autolab, refer to the Piazza posts.

To get started, you can log into the Autolab website (https://autolab.andrew.cmu.edu). From there you should see 10-401 in your list of courses. Download the template for Homework 1 and extract the contents (i.e., by executing `tar xvf hw1.tar` at the command line). In the archive you will find one `.m` file for each of the functions that you are asked to implement and a file that contains the data for this problem, `HW1Data.mat`. To finish each programming part of this problem, open the corresponding `.m` file and complete the function defined in that file. To submit your solutions, create a new tar archive of the top-level directory (i.e., by executing `tar cvf hw1.tar hw1`) and upload that through the Autolab website.

The file `HW1Data.mat` contains the data that you will use in this problem. You can load it from Octave by executing `load('HW1Data.mat')` in the Octave interpreter. 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 0 if the i^{th} document belongs to The Economist and 1 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. These are the data you will test your classifier on and should not be used for training.
- Finally, `XTrainSmall` and `yTrainSmall` are subsets of `XTrain` and `yTrain` which are used in (g).

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: $\log(p(x) * p(y)) = log(p(x)) + log(p(y))$.

(a) [**4 Points**] Complete the function `logProd(x)` which 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., $\texttt{logProd}(\texttt{x}) = \log\left(\prod_i p_i\right)$.

Training Naive Bayes

(b) [**13 Points**] Complete the function `[D] = NB_XGivenY(XTrain, yTrain)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{1, \dots, V\}$ and class index $y \in \{0, 1\}$, the entry `D(y,w)` is the MAP estimate of $\theta_{yw} = P(X_w = 1|Y = y)$ with a Beta(2,1) prior distribution.

(c) [**8 Points**] Complete the function `[p] = NB_YPrior(yTrain)`. The output `p` is the MLE for $\rho = P(Y = 1)$.

(d) [**15 Points**] Complete the function `[yHat] = NB_Classify(D, p, X)`. The input `X` is an $n \times V$ matrix containing n feature vectors (stored as its rows). The output `yHat` is an $n \times 1$ 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.]

(e) [**4 Points**] Complete the function `[error] = ClassificationError(yHat, yTruth)`, which takes two vectors of equal length and returns the proportion of entries that they disagree on.

Questions

You will answer these 8 questions on QNA at http://bit.ly/HW1_qna_NB. Each question sub-part is worth 1 point if answered correctly on the first try and 1/2 point if answered correctly after more than one attempt.

(f) [**1 Point**] 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);
```

(1) How do the train and test errors compare? Explain any significant differences.

(g) [**2 Points**] Repeat the steps from part (f), 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 part (f). [Hint: When we have less training data, does the prior have more or less impact on our classifier?]

(1) Explain any differences between the train errors in this question and in part (f).

(2) Explain any differences between the test errors in this question and in part (f).

(h) [**5 Points**] Finally, we will try to interpret the learned parameters. Train your classifier on the data contained in `XTrain` and `yTrain`. For each class label $y \in \{0, 1\}$, list the six words that the model says are most likely to occur in a document from class y. Also for each class label $y \in \{0, 1\}$, list the six words w that maximize the following quantity:

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

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".)

(1) Give one word in the top eight highest probabilities $P(X_w = 1|Y = 0)$ (*i.e.*, words most likely to appear in Economist articles).

(2) Give one word in the top eight highest probabilities $P(X_w = 1|Y = 1)$ (*i.e.*, words most likely to appear in Onion articles).

(3) Give one word in the top eight that maximize the quantity in Equation 1 for $y = 0$.

(4) Give one word in the top eight that maximize the quantity in Equation 1 for $y = 1$.

(5) Which list of words describes the two classes better?