10-601 Machine Learning: Homework 2

Due 5:30 p.m. Thursday, February 4, 2016

Instructions

- **Late homework policy:** Homework is worth full credit if submitted before the due date, half credit during the next 48 hours, and zero credit after that. You *must* turn in at least $n-1$ of the n homeworks to pass the class, even if for zero credit.

- **Collaboration policy:** Homeworks must be done individually, except where otherwise noted in the assignments. "Individually" means each student must hand in their own answers, and each student must write and use their own code in the programming parts of the assignment. It is acceptable for students to collaborate in figuring out answers and to help each other solve the problems, though you must in the end write up your own solutions individually, and you must list the names of students you discussed this with. We will be assuming that, as participants in a graduate course, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

- **Submission:** You must submit your solutions on time BOTH electronically by submitting to autolab AND by dropping off a hardcopy in the bin outside Gates 8221 by 5:30 p.m. Thursday, February 4, 2016. We recommend that you use LaTeX, but we will accept scanned solutions as well. On the Homework 2 autolab page, you can click on the "download handout" link to download the submission template, which is a tar archive containing a blank placeholder pdf for your written solutions, and an Octave `.m` file for each programming question. Replace each of these files with your solutions for the corresponding problem, create a new tar archive of the top-level directory, and submit your archived solutions online by clicking the "Submit File" button.

  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: Independent events and Bayes Theorem

(a) [**5 Points**] For events A, B prove:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}.$$

(b) [**5 Points**] Let $A_1, ..., A_n$ be a partition of the sample space Ω, prove that for any event B,

$$P(B) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

Definition: A partition of the sample space Ω is a sequence of disjoint events $A_1, A_2, ..., A_n$ such that $\bigcup_{i=1}^{n} A_i = \Omega$.

(c) [**5 Points**] Let $A_1, ..., A_n$ be a partition of the sample space Ω such that $P(A_i) > 0$ for all $i = 1, ...n$. Prove that if $P(B) > 0$, then for each $i = 1, ..., n$,

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_k P(B|A_k)P(A_k)}.$$

(d) **[5 Points]** Let A, B, and C be any events. Which of the following statements are true? Justify your responses (e.g. *True, this is a direct implication of Bayes Theorem* or *False, it is only true if P(A)=P(B)*. Alternatively, you can provide a counterexample to justify that something is false).

(1) $P(A,B,C) = P(A|B,C)P(B|C)P(C)$

(2) $P(A,B) = P(A|B)P(B|A)$

(3) $P(A,B,C) = P(B|A,C)P(C,A)$

(4) $P(A,B,C) = P(B|A,C)P(C,A)P(C)$

(5) $P(A,B) = P(A)P(B)$

(e) **[5 Points]** Let A be any event, and let X be a random variable defined by

$$X = \begin{cases} -1 & \text{if event } A \text{ occurs} \\ 0 & \text{otherwise.} \end{cases}$$

Show that $P(A) + \mathbb{E}[X] = 0$, where $\mathbb{E}[X]$ denotes the *expected value* of X. Does this result still hold if the value of X is 1 when event A occurs, instead of -1?

Problem 2: Maximum Likelihood Estimation

This problem explores maximum likelihood estimation, which is a technique for estimating an unknown parameter of a probability distribution based on observed samples. Suppose we observe the values of n iid[1] random variables $X_1, \ldots, X_n$ drawn from a single Geometric distribution with parameter θ. In other words, for each X_i and natural number k, we know

$$P(X_i = k) = (1-\theta)^k \theta$$

Our goal is to estimate the value of θ from these observed values of X_1 through X_n.

For any hypothetical value $\hat{\theta}$, we can compute the probability of observing the outcome $X_1, \ldots, X_n$ if the true parameter value θ were equal to $\hat{\theta}$. This probability of the observed data is often called the *data likelihood*, and the function $L(\hat{\theta}) = P(X_1, \ldots, X_n|\hat{\theta})$ that maps each $\hat{\theta}$ to the corresponding likelihood is called the *likelihood function*. A natural way to estimate the unknown parameter θ is to choose the $\hat{\theta}$ that maximizes the likelihood function. Formally,

$$\hat{\theta}^{\text{MLE}} = \operatorname*{argmax}_{\hat{\theta}} L(\hat{\theta}).$$

Often it is more convenient to work with the log likelihood function $\ell(\hat{\theta}) = \log L(\hat{\theta})$. Since the log function is increasing, we also have

$$\hat{\theta}^{\text{MLE}} = \operatorname*{argmax}_{\hat{\theta}} \ell(\hat{\theta}).$$

(a) **[5 Points]** Write a formula for the log likelihood function, $\ell(\hat{\theta})$. Your function should depend on the random variables $X_1, \ldots, X_n$, the hypothetical parameter $\hat{\theta}$, and should be simplified as far as possible (i.e., don't just write the definition of the log likelihood function). Does the log likelihood function depend on the order of the random variables?

(b) **[5 Points]** Consider the following sequence of 15 samples:

$$X = (1, 0, 3, 5, 18, 14, 5, 7, 13, 9, 0, 17, 4, 24, 3).$$

Write a short computer program that plots three log likelihood functions, one based on the first 5 samples in X (i.e., $(1, 0, 3, 5, 18)$), one based on the first 10 samples, and one based on all 15 samples. Each plot

[1] iid means Independent, Identically Distributed.

should show the log likelihood of each value of $\hat{\theta}$ in $\{0.01, 0.02, \ldots, 0.5\}$. The x-axis should be $\hat{\theta}$ and the y-axis should be $\ell(\hat{\theta})$. For each plot, mark the location of the maximum likelihood estimator. Please include both the plots and the code used to generate them (but, since this code is not autograded, please just include it in your written submission).

(c) [**5 Points**] Derive a closed form expression for the maximum likelihood estimate (hint: recall that if x^* maximizes $f(x)$, then $f'(x^*) = 0$). Does your closed form expression agree with the plots?

(d) [**5 Points**] Briefly explain (in 1-2 sentences) why the log likelihood functions become more negative as the number of samples increases.

Problem 3: 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 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 1 for The Economist or 2 for The Onion. Later in the question we give instructions for loading this data into Octave.

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,

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

(a) [**5 points**] Prove the classification rule can be rewritten as

$$\hat{Y} = \operatorname*{argmax}_y \left(\prod_{w=1}^{V} P(X_w | Y = y) \right) P(Y = y).$$

(b) [**5 points**] How many parameters are needed to represent the distribution $P(X|Y = y)$ when using the Naive Bayes assumption? How many are needed if we do not use the Naive Bayes assumption? Based on this difference, in which cases is there a big gain from making this assumption?

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

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

For both The Economist and The Onion, we believe that each word w has a non-zero chance of appearing, but it is more likely that w will not occur in any particular document. We incorporate this belief by computing a MAP estimate using a $\text{Beta}(1.001, 1.9)$ prior on θ_{wy}. This has the added benefit of ensuring that none of our estimates of θ_{wy} are equal to 0 or 1 (which can cause problems for Naive Bayes).

Similarly, the distribution of Y (when we consider it alone) is a Bernoulli distribution (except taking values 1 and 2 instead of 0 and 1) with parameter ρ. In other words, there is some unknown number ρ such that

$$P(Y = 1) = \rho \quad \text{and} \quad P(Y = 2) = 1 - \rho.$$

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.

Programming Instructions

Parts (c) through (e) of this question each ask you to implement one function related to the Naive Bayes classifier. You will submit your code online through the CMU autolab system, which will execute it remotely against a suite of tests. Your grade will be automatically determined from the testing results. Since you get immediate feedback after submitting your code and you are allowed to submit as many different versions as you like (without any penalty), it easy for you to check your code as you go.

Our autograder requires that you write your code in Octave. Octave is a free scientific programming language with syntax identical to that of MATLAB. Installation instructions can be found on the Octave website (`http://www.gnu.org/software/octave/`), and we have posted links to several Octave and MATLAB tutorials on Piazza.

To get started, you can log into the autolab website (`https://autolab.andrew.cmu.edu`). From there you should see 10-601B in your list of courses. Download the template for Homework 2 and extract the contents (i.e., by executing `tar xvf hw2.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, `HW2Data.mat`. To finish each programming part of this problem, open the corresponding `.m` file and complete the function defined in that file. When you are ready to submit you solutions, you will create a new tar archive of the top-level directory (i.e., by executing `tar cvf hw2.tar hw2`) and upload that through the Autolab website.

The file `HW2Data.mat` contains the data that you will use in this problem. You can load it from Octave by executing `load("HW2Data.mat")` in the octave interpreter. After loading the data, you will see that there are 5 variables: `Vocabulary`, `XTrain`, `yTrain`, `XTest`, and `yTest`.

- `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.
- Finally, `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.

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

We provide the function `logProd(x)` so you can use it in your implementation. This function 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

(c) [**8 Points**] Complete the function `[D] = NB_XGivenY(XTrain, yTrain)`. The output `D` is a $2 \times V$ matrix, where for any word index $w \in \{1, \ldots, V\}$ and class index $y \in \{1, 2\}$, the entry `D(y,w)` is the MAP estimate of $\theta_{yw} = P(X_w = 1|Y = y)$ with a Beta(1.001,1.9) prior distribution.

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

(e) [**8 Points**] 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.]

Evaluating Naive Bayes

To help you evaluate your results, we provide the function `[error] = ClassificationError(yHat, yTruth)`, which takes two vectors of equal length and returns the proportion of entries that they disagree on.

Questions

(f) [**5 Points**] 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? Which is more representative of the error we would expect to have on a new collection of articles? Does Naive Bayes attempt to minimize the training error?

(g) [**8 Points**] In this question we explore how the size of the training data set affects the test and train error. For each value of m in $\{100, 130, 160, \ldots, 580\}$, train your Naive Bayes classifier on the first m training examples (that is, use the data given by `XTrain(1:m,:)` and `yTrain(1:m)`). Plot the training and testing error for each such value of m. The x-axis of your plot should be m, the y-axis should be error, and there should be one curve for training error and one curve for testing error. Explain the general trend of both the training and testing error curves.

(h) [**8 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 $y \in \{1, 2\}$, create three lists according to the following criteria (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"):

- Top five words that the model says are most likely to occur in a document from class y. That is, the top five words according to this metric:

$$P(X_w = 1|Y = y)$$

- Top five words w according to this metric:

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

- Top five words w according to this metric:

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

Which list of words is more informative about the class y? Briefly explain your reasoning.