Will Monroe
CS 109

Problem Set #6
August 10, 2017

Problem Set #6
Due: 12:30pm on Wednesday, August 16th

With problems by Mehran Sahami and Chris Piech

Please note that there will be **no late days allowed on this assignment**.

For each problem, briefly explain/justify how you obtained your answer. Brief explanations of your answer are necessary to get full credit for a problem even if you have the correct numerical answer. The explanations help us determine your understanding of the problem whether or not you got the correct answer. Moreover, in the event of an incorrect answer, we can still try to give you partial credit based on the explanation you provide. It is fine for your answers to include summations, products, factorials, exponentials, or combinations; you don't need to calculate those all out to get a single numeric answer.

Unless otherwise stated, you may also use functions in a library like Python's `scipy.stats` to compute values of PMFs and CDFs; if you use these, provide your code that calls these functions and explain how you arrived at each parameter to a function or constructor.

Written Problems

1. Program A will run 20 algorithms in sequence, with the running time for each algorithm being independent random variables with mean = 50 seconds and variance = 100 seconds^2. Program B will run 20 algorithms in sequence, with the running time for each algorithm being independent random variables with mean = 52 seconds and variance = 200 seconds^2.
    a. What is the approximate probability that Program A completes in less than 950 seconds?
    b. What is the approximate probability that Program B completes in less than 950 seconds?
    c. What is the approximate probability that Program A completes in less time than Program B?

2. A fair 6-sided die is repeatedly rolled until the total sum of all the rolls exceeds 100. Approximate the probability that *at least* 30 rolls are necessary to reach a sum that exceeds 100.

3. From past experience, we know that the midterm score for a student in CS 106Z is a random variable with mean = 70. Assume that exam scores can be real values (i.e., fractional points can be given), but scores cannot be negative.
    a. Give an upper bound for the probability that a student's midterm score will be greater than or equal to 80.
    b. Now, say we are given the additional information that the variance of a student's midterm exam score in CS 106Z is 20 (and you can use this information for parts (c) and (d) below as well). Give a bound on the probability that a student's midterm score is between 60 and 80, inclusive.

c. According to Chebyshev's inequality, how many students would have to take the midterm in order to ensure, with at least 90% probability, that the class average would be within 5 of 70?
d. According to the Central Limit Theorem, how many students would have to take the midterm in order to ensure, with at least 90% probability, that the class average would be within 5 of 70?

4. Consider a sample of I.I.D. exponential random variables $X_1, X_2, \ldots, X_n$, where each $X_i \sim \text{Exp}(\lambda)$.
   a. Derive the maximum likelihood estimate for the parameter λ in the Exponential distribution.
   b. Is the estimator you derived in part (a) unbiased? You should give a "yes" or "no" answer and a short informal justification for your answer. A formal derivation/proof is not needed. (Hint: Johan Jensen might be interested in your answer).
   c. Is the estimator you derived in part (a) consistent? (Again, you should give a "yes" or "no" answer and a short informal justification for your answer. A formal derivation/proof is not needed).

5. Say you have a set of binary input features/variables $X_1, X_2, \ldots, X_m$ that can be used to make a prediction about a discrete binary output variable Y (i.e., each of the X_i as well as Y can only take on the values 0 or 1). In using the input features/variables $X_1, X_2, \ldots, X_m$ to make a prediction about Y, recall that the Naïve Bayes classifier makes the simplifying assumption that $P(X_1, X_2, \ldots, X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)$ in order to make it tractable to compute $\arg\max_Y P(\mathbf{X}, Y) = \arg\max_Y P(X_1, X_2, \ldots, X_n \mid Y)P(Y)$. Say that the first k input variables $X_1, X_2, \ldots, X_k$ are actually all identical copies of each other, so that when one has the value 0 or 1, they all do. Explain informally, but precisely, why this may be problematic for the model learned by the Naïve Bayes classifier.

Final coding problem

For the following problem, you will be implementing the Naïve Bayes classifier. You must implement your algorithm in your choice of C, C++, Java, or Python. Please note that other programming languages (e.g., R, MATLAB, etc.) are not allowed on this assignment. You can feel free (but are under no obligation) to use the CS106A ACM Java libraries, the CS106B/X C++ libraries, or the C++ Standard Template Libraries (STL) as well as the standard libraries that are part of these languages. You should not use any non-standard libraries for these languages (e.g., other libraries from the web). As a special exception, if you use Python, you may use the libraries NumPy and SciPy, in addition to the libraries that are standard in the language (e.g., math, string, collections, etc. libraries). Using NumPy and SciPy is not required; the assignment can be completed with only standard math operations. No other external libraries are allowed (e.g., scikit-learn or Tensorflow). You should turn in your source code as well as answers to the questions listed below. It's fine if your implementation is in a single file or if you use multiple files. In either case, please provide any code you write.

You will be testing the algorithm with three datasets. A description of the datasets you will be using, the file format for the data files, and instructions on how to obtain the data files are given

below. Note: you do not need to do any error checking in your file reading code (you can assume the data is always correctly formatted). To simplify your implementation, you can assume that all input features are always binary variables (0 or 1), and the output class is also always a binary variable (0 or 1). For the assignment, our main interest is the results you obtain with the learning algorithm. As a result, you do not need to worry about the generality of your implementation—i.e., you can write your algorithms to only deal with binary input/output features. Your code should, however, be general enough to work for any number of input features or data instances (within reason), as the different datasets you will be dealing with contain different numbers of input features and data instances. We will be grading your code only on functionality, not on programming style. With that said, it is still in your interest to write good modular code as there are many opportunities for code reuse in implementing this assignment.

Datasets

You will be running your learning algorithm on three datasets (each of which has a respective training data file and testing data file). The datasets are described in more detail below. The data files are available on the companion page to this assignment: `http://cs109.stanford.edu/psets/pset6.html`.

Simple (`simple-train.txt`, `simple-test.txt`)

This is a simple dataset provided primarily to help you determine that your code is working correctly. There are two input features, and the output class value is determined by the value of the first feature (i.e., $y = x_1$). The training dataset and testing dataset are identical, each containing four data vectors. Your Naïve Bayes classifier implementation should be able to classify all instances in the simple testing dataset with 100% accuracy after training on the simple training set.

Congressional voting records (`vote-train.txt`, `vote-test.txt`)

This dataset contains the congressional voting records from the U.S. House of Representatives in 1984 on several key issues. Each input vector represents the voting record for one member of Congress. There are 48 binary input features. The output class value represents the political affiliation of the Congressperson (Democrat or Republican, encoded in binary). The training dataset contains 300 data vectors, and the testing dataset contains 135 data vectors.

(Thanks to Jeff Schlimmer for providing this data to the UC Irvine Machine Learning Data Repository.)

Heart tomography diagnosis (`heart-train.txt`, `heart-test.txt`)

This dataset contains data related to diagnosing heart abnormalities based on tomography (X-ray) information. Each input vector represents data extracted from the X-ray of one patient's heart. There are 22 binary input features. The output class value represents the diagnosis of the patient's heart (normal or abnormal, encoded in binary). The training dataset contains 80 data vectors, and the testing dataset contains 187 data vectors.

(Thanks to Lukasz Kurgan and Krzysztof Cios for providing this data to the UC Irvine Machine Learning Data Repository.)

Data file format

All the data files described above adhere to the following file format:

```
<number of input variables per vector>
<number of data vectors in file>
<first data vector>
<second data vector>
...
<n-th data vector>
```

Note that each data vector in the file consists of a number of input variable values that are binary (0 or 1). The input variable values are separated by a single space. The last input variable value is immediately followed by a colon character ':', then a single space and then the value of the binary output variable for the vector.

For example, here is the annotated `simple-train.txt` data file (with annotations in italic font on the right-hand side):

File: `simple-train.txt`	Explanation of lines in data file
`2`	← There are 2 input variables per vector in the file
`4`	← There are 4 data vectors in the file
`0 0: 0`	← First data vector (has class 0)
`0 1: 0`	← Second data vector (has class 0)
`1 0: 1`	← Third data vector (has class 1)
`1 1: 1`	← Fourth data vector (has class 1)

Training and testing your algorithm

The "training" data files should be used to train your learning algorithm (i.e., determine the model parameters). The "testing" data file should be used to determine the accuracy of your model after the training phase is complete. In other words, when we describe training an algorithm below, you should take that to mean that you are working only with the "-train" file for a particular dataset to determine the parameters of your model. When we then describe testing a model you should take that to mean that you are using only the "-test" file for a particular dataset to determine how well your model does at classifying the data.

Measuring model accuracy

After a model is trained, we determine its accuracy by testing it on a new set of data (generally not the same data we used to train the model). We measure the model's accuracy by determining how many of the testing vectors were correctly classified — that is, the number of times the output class value predicted by the model was the same as the actual output class value provided in the data. We report accuracy by indicating the number testing data vectors that were tested of each class, and the number that were correctly classified. For example, say we have a testing dataset consisting of 12 vectors total, where the first 5 vectors are of class y = 0 and the remaining 7 of class y = 1. When

we then make predictions for each data vector using our model, say we correctly predict class = 0 for 4 out of the first 5 vectors and then correctly predict class= 1 for 5 out of the next 7 vectors. Our overall accuracy for the model would be 0.75 since we correctly classified a total of 9 out of 12 vectors. We would report these results as follows:

```
Class 0: tested 5, correctly classified 4
Class 1: tested 7, correctly classified 5
Overall: tested 12, correctly classified 9
Accuracy = 0.750
```

You should use this same accuracy reporting scheme for your Naïve Bayes implementation.

6. Implement the Naïve Bayes classifier for binary input/output data. Specifically, your classifier should make predictions for the output variable using the rule: $\hat{Y} = \arg\max_y P(\mathbf{X} \mid Y)P(Y)$, by employing the Naïve Bayes assumption, which states that:

$$P(\mathbf{X} \mid Y) = P(X_1, X_2, \ldots, X_m \mid Y) = \prod_{i=1}^{m} P(X_i \mid Y)$$

   Thus, your program will need to estimate the values $P(Y)$ as well as $P(X_i \mid Y)$ for all $1 \leq i \leq m$ from the training data. Note that to estimate the probability mass function $P(X_i \mid Y)$, you will need to estimate both $P(X_i \mid Y = 0)$ and $P(X_i \mid Y = 1)$. For each of parts (a)-(c) below, you should run your algorithm twice, the first time computing your probability estimates using maximum likelihood estimation and the second time computing your probability estimates using Laplace estimation.

   a. Train your algorithm on the data file `simple-train.txt`. Test your algorithm on the data file `simple-test.txt` and report your classification accuracy. Remember to do this once with maximum likelihood estimation and once with Laplace estimation. As a sanity check, you should be able to achieve 100% classification accuracy on the testing data using a model trained with maximum likelihood estimation.

   b. Train your algorithm on the data file `vote-train.txt`. Test your algorithm on the data file `vote-test.txt` and report your classification accuracy. Remember to do this once with maximum likelihood estimation and once with Laplace estimation. To give you a sanity check of how well you should be doing, you should be able to achieve at least 90% classification accuracy on the testing data using a model trained with maximum likelihood estimation.

   c. Train your algorithm on the data file `heart-train.txt`. Test your algorithm on the data file `heart-test.txt` and report your classification accuracy. Again, remember to do this once with maximum likelihood estimation and once with Laplace estimation.

d. After running your algorithms on both the vote and heart data, did you see any difference between using maximum likelihood estimation versus Laplace estimation in your accuracy results? In general, under what conditions (i.e., characteristics of the datasets) do you think using Laplace estimation (rather than maximum likelihood estimation) would be better (e.g., likely improve classification accuracy)? Under what conditions do you think using maximum likelihood estimation (rather than Laplace estimation) would be better?

e. Include your code for the Naïve Bayes classifier in your Gradescope submission. Please make sure lines are not cut off at the right or bottom edge of pages.