

Chris Piech  
CS 109

Pset #6  
Nov 16<sup>th</sup>, 2021

## Problem Set #6

### Due: 11:59pm on Wednesday, Dec 1st

---

**For each problem, explain/justify how you obtained your answer** in order to obtain full credit. In fact, most of the credit for each problem will be given for the derivation/model used as opposed to the final answer. Make sure to describe the distribution and parameter values you used, where appropriate. Provide a numeric answer for all questions when possible.

### Written problems

1. Consider a sample of I.I.D. random variables  $X^{(1)}, X^{(2)}, \dots, X^{(n)}$ , where each  $X^{(i)} \sim \text{Exp}(\lambda)$ . Derive the maximum likelihood estimate for the parameter  $\lambda$  in the Exponential distribution.

Answer:

2. Say you have a set of binary input features/variables  $X_1, X_2, \dots, X_m$  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). Say that the first  $k$  input variables  $X_1, X_2, \dots, X_k$  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 Bayesian Classifier.

Answer:

## Programming problems

### *Requirements*

For the following problems, you will be implementing two learning algorithms: the Naïve Bayesian Classifier and Logistic Regression. You may implement your algorithms any language, though we will only be offering programming help for python. You can feel free (but are under no obligation) to use standard libraries. You should **not** use any library that actually implements machine learning algorithms. For each algorithm, **you should turn in your source code as well as answers to the questions listed below**. It's fine if your implementation of both algorithms is in a single file or if you use multiple files. In either case, please provide any code you write.

For each algorithm you write, you will be testing it with three data sets. A description of the data sets 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 that you understand how the machine learning algorithms work. 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 algorithms on four data sets (each of which has a respective training data file and testing data file). **See the respective readme files for more details.**

### **Simple (`simple-train.csv`, `simple-test.csv`)**

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 data set and testing data set are identical, each containing four data vectors. Both your Naïve Bayesian Classifier and Logistic Regression implementations should be able to classify all instances in the simple testing data set with 100% accuracy after training on the simple training set.

### **Heart tomography diagnosis (`heart-train.csv`, `heart-test.csv`)**

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 data set contains 80 data vectors, and the testing data set contains 187 data vectors.

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

### **Genetic ancestry (`ancestry-train.csv`, `ancestry-test.csv`)**

This dataset contains DNA nucleotide readings from 467 individuals. Each input vector represents locations in the human genome and whether the individual's nucleotide at given locations matches the human reference genome. The output class value represents the super population of the user.

(Thanks to Jim Notwell and Gill Bejerano for providing this dataset.)

### **Netflix dataset (`netflix-train.csv`, `netflix-test.csv`)**

This dataset contains real user ratings on Netflix.com. Each input vector represents ratings by a single user for the 30 most commonly rated movies (1 = rating of 5). The output class value represents whether the user rated the target movie (Love Actually) as a 5.

(This dataset is made available by Reed Hastings with processing by Chris Piech.)

## **Data file format**

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

```
<header >
<first data point>
<second data point >
...
<n-th data point >
```

Note that each data vector in the file is comprised of a number of input variable values that are binary (0 or 1). The input variable values are separated by a single comma. The last value is always the binary output variable that you are meant to learn to predict. For example, here is the annotated **simple-train.csv** data file (with annotations in italic font on the right-hand side):

| File: simple-train.csv | Explanation of lines in data file                       |
|------------------------|---|
| x1, x2, Label          | -> each data point has two features and a single output |
| 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)                     |

## Demographics

Some datasets (Netflix and Heart) have demographic variables. These should **not** be used in questions 1 and 2, but rather will be used to analyze your results in question 3.

## How to get data sets

The data files are available on the CS109 website home page: <http://cs109.stanford.edu>.

## Actual programming problems

### Training and Testing your algorithms

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 of data vectors that were tested of each class, and the number that were correctly classified. For example, say we have a testing data set 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  $\hat{y} = 0$  for 4 out of the first 5 vectors and then correctly predict  $\hat{y} = 1$  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 = 0.75
```

You should use this same accuracy reporting scheme for the algorithms you implement below.

1. Implement the Naïve Bayesian Classifier for binary input/output data. Specifically, your classifier should make predictions for the output variable using the rule:

$$\hat{Y} = \arg \max_y P(X|Y)P(Y)$$

by employing the Naïve Bayes assumption, which states that:

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

Thus, your program will need to estimate the values  $P(Y)$  as well as  $P(X_i|Y)$  for all  $1 \leq i \leq m$  from the training data. Note that to estimate the probability mass function  $P(X_i|Y)$ , you will need to estimate both  $P(X_i|Y = 0)$  and  $P(X_i|Y = 1)$ .

- a. Train your algorithm on the data file **simple-train.csv**. Test your algorithm on the data file **simple-test.csv** and report your classification accuracy. Run your algorithm once with Maximum Likelihood Estimators (MLE) and once with Laplace Estimators. As a sanity check, you should be able to achieve 100% classification accuracy on the testing data using a model trained with Maximum Likelihood Estimators. You do not have to submit anything for this problem.

Answer:

- b. Train your algorithm on the data file **netflix-train.csv**. Test your algorithm on the data file **netflix-test.csv**. Run your algorithm twice, once with MLE and once with Laplace Estimators. Note that even though the data files contain a column for demographics, you should NOT use those values in training.

To help you calibrate how well your algorithm is working, here are two examples of accuracy you should be getting (NOTE that this was trained WITH demographic column, but your answer should NOT train with the demographic column)

- Training accuracy: 0.7175555555555555
- Testing accuracy: 0.67
- Model: Naïve Bayes
- Dataset: netflix WITH demographic included
- Hyperparameters: use\_laplace\_add\_one = False

For both versions of Naïve Bayes, answer the following questions:

- i. What is your estimate for  $P(Y = 1)$ ?
- ii. For all values of  $i$ , what is your estimate  $P(X_i = 1|Y = 1)$ ?
- iii. For all values of  $i$ , what is your estimate  $P(X_i = 1|Y = 0)$ ?
- iv. Report your classification accuracy for both MLE and Laplace estimators.

Answer:

- c. For Naïve Bayes trained on **netflix-train.csv** with an MLE estimator answer the following questions:

- i. Using the probabilities that you estimated for Naïve Bayes, decide which five movies are the most indicative that a user will like Love Actually. In other words, for which five movies is the ratio  $P(Y = 1|X_i = 1)/P(Y = 1|X_i = 0)$  the largest.
- ii. Give one example of a miss-prediction. Try and explain what went wrong.

Answer:

- d. Train your algorithm on the data file **ancestry-train.csv**. Test your algorithm on the data file **ancestry-test.csv** and report your classification accuracy. Again, remember to do this once with Maximum Likelihood Estimators and once with Laplace Estimators.

Answer:

- e. Train your algorithm on the data file **heart-train.csv**. Test your algorithm on the data file **heart-test.csv** and report your classification accuracy. Again, remember to do this once with Maximum Likelihood Estimators and once with Laplace Estimators.

Answer:

- f. Include a copy of your code for Naïve Bayes.

Answer:

2. Implement Logistic Regression for binary input/output data. Specifically, you should implement the gradient ascent algorithm described in class.

Note that you will need to use an exponential function ( $e^x$ ) to implement Logistic Regression. In Python, you will find the function `np.exp(x)` from the library `numpy` helpful.

- a. Train your algorithm on the data file **simple-train.csv**. Use learning rate  $\eta = 0.0001$  and 10,000 training steps. Test your algorithm on the data file **simple-test.csv** and report your classification accuracy. You should be able to achieve 100% classification accuracy on the testing data.

Answer:

- b. Train your algorithm on the data file **netflix-train.csv**. Use learning rate  $\eta = 0.0001$  and 3,000 training steps. Note that even though the data files contain a column for demographics, you should not use those values in training

Test your algorithm on the data file **netflix-test.csv**:

- i. Report your classification accuracy.
- ii. According to the weights of your logistic regression function, which five movies are the strongest predictors that a user will like Love Actually?
- iii. What is the log likelihood of the training data when all the parameters are 0?
- iv. What is the log likelihood of the training data after training?

Answer:

- c. Train your algorithm on the data file **ancestry-train.csv**. Use learning rate  $\eta = 0.0001$  and 10,000 training steps. Test your algorithm on the data file **ancestry-test.csv** and report your classification accuracy.

To help you calibrate how well your algorithm is working, here are two examples of accuracy you should be getting.

- Training accuracy: 0.8869257950530035
- Testing accuracy: 0.8369565217391305
- Model: Logistic Regression
- Dataset: ancestry (there is no demographic variable)
- Hyperparameters: `learning_rate = 0.01`, `max_steps = 100`

Answer:

- d. Train your algorithm on the data file **heart-train.csv**. Experiment with using different learning rates  $\eta$ , where you are still testing your algorithm on the data file **heart-test.csv**. Each time use 10,000 training steps. As a starting point for experimenting, try  $\eta = 0.0005$  and  $\eta = 0.00002$  (i.e., values for  $\eta$  that are larger and smaller than 0.0001 by a factor of 5), and then continue experimenting from there. Report the highest classification accuracy you could obtain on the testing data, and what was the value of the learning rate  $\eta$  you used to obtain it. Explain why do you think the learning rate had such an effect on your classification accuracy.

Answer:

- e. Include a copy of your code for Logistic Regression.

Answer:

3. Analysis of Fairness. Two of the datasets (Netflix and Heart) have a column named Demographics. This column represents a protected demographic variable (we don't specify exactly which demographic, but you could imagine it is a "protected variable" like race, gender, age, etc). We can use this variable to analyze "fairness". For this question we are going to focus our analysis on Naive Bayes:

- a. Imagine a researcher trains a deep learning algorithm on Heart and Netflix. They can optionally include the demographic variable. Leaving out a protected attribute is one way to create "fairness through unawareness," a form of procedural fairness. But in both datasets, when they include the demographic variable, accuracy goes up for all demographics.
- Do you think that demographic input *should* be used as an *input feature* for Heart? How about for Netflix? Assume no additional corrective action is taken to achieve a standard of fairness. Justify your choice in 2-3 sentences.
  - If you do not include the demographic variable, is it still possible for the model to make its decision based on the demographic? Explain in 2-3 sentences.

Answer:

- b. Does the trained Naive Bayes algorithm satisfy **parity** on the heart dataset with respect to the demographic variable? What about the Netflix dataset?

Answer:

- c. Does the trained Naive Bayes algorithm satisfy **calibration** on the heart dataset? What about the Netflix dataset?

Answer:



- d. In a medical context, a false negative often leads to delayed treatment, which could be dangerous, while a false positive often leads to over-treatment, which could increase the likelihood of unnecessary medical treatments and complications. Given these possible consequences, which form of fairness should we strive to achieve in the classifier we train on the Heart dataset and why? Justify your choice in 2-3 sentences.

Answer:

4. Extra-Credit: Modify the Logistic Regression algorithm to learn to be both be calibrated and make accurate predictions. Verify your answer works on the heart dataset.