Lisa Yan
CS109

Problem Set #6
November 15, 2019

Problem Set #6
Due: 1:00pm on Wednesday, December 4th

Note: The last day this assignment will be accepted (late) is Friday, December 6th.

With problems by Mehran Sahami and Chris Piech

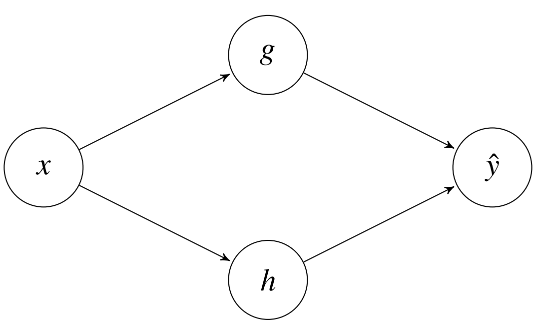
For each problem, briefly explain/justify how you obtained your answer in order to obtain full credit. In fact, most of the credit for each problem will be given for the derivation/model used as opposed to the final answer. Make sure to describe the distribution and parameter values you used where appropriate.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

Written Problems

1. Consider the Exponential distribution. It is your friend . . . really. Specifically, 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) biased? 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).

2. 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). 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.

3. Consider this 4-neuron neural network that is trained on n datapoints $(x^{(i)}, y^{(i)})$:



$$g = \text{sigmoid}(\theta_1 \cdot x)$$
$$h = \text{sigmoid}(\theta_2 \cdot x)$$
$$\hat{y} = \text{sigmoid}(\theta_3 \cdot g + \theta_4 \cdot h)$$
$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

   a. Calculate the gradients of the log-likelihood function with respect to all four parameters.
   b. Explain in a few sentences how you could use the function that you calculated in part (a) to train the neural network.

Programming Problems

For the following problems, you will be implementing two learning algorithms: the Naïve Bayes classifier and Logistic Regression. You must implement your algorithm any language, though we will only be offering programming help for Python. You should free (but are under no obligation) to use standard libraries like `NumPy` and `SciPy`. You should **not** use any library that actually implements machine learning algorithms (e.g., scikit-learn or Tensorflow).

Submission

For each algorithm, **you should turn in your source code as well as answers to the questions listed below**. Note that you need to submit to **two** Gradescope assignments for this problem set: Written questions to all answers should be submitted in a PDF through the "Problem Set #6" Gradescope assignment. Only your code should be submitted through the "Problem Set #6 Code" Gradescope assignment. Both Gradescope assignments have the same deadline.

You will not include code in the PDF directly. Instead, please place all of your files of code (and only those files) into a zip (with any name you'd like) and upload that zip to Gradescope in the "Problem Set #6 Code" assignment. Once uploaded, click into the "code" tab and make sure that all of your code files are there.

Implementation details

- For each algorithm you write, you will be testing it with four 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.
- 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 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 this 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 datasets (each of which has a respective training data file and testing data file). **See the respective README files for more details.**

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. Both your Naïve Bayes classifier and Logistic Regression implementations should be able to classify all instances in the simple testing dataset with 100% accuracy after training on the simple training set.

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 Repository.)

Genetic ancestry (`ancestry-train.txt`, `ancestry-test.txt`)

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.txt`, `netflix-test.txt`)

This dataset contains real user ratings from Netflix. 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.

(Thanks to Reed Hastings for providing this dataset, with processing by Chris Piech.)

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)

Actual programming problems

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 classification 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 $\hat{y} = 0$ for 4 out of the first 5 vectors and then correctly predict class $\hat{y} = 1$ 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.

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

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

Thus, your program will need to estimate the values $P(Y)$ as well as $P(X_i \mid Y)$ for all $1 \le i \le m$ 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 $P(X_i \mid Y = 1)$.

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. Run your algorithm twice—once with Maximum Likelihood Estimators (MLE) and once with Laplace Estimatiors. As a sanity check, you should be able to achieve 100% classification accuracy on the testing data using a model trained with MLE.

b. Train your algorithm on the data file `netflix-train.txt`. Test your algorithm on the data file `netflix-test.txt`. Run your algorithm twice—once with MLE and once with Laplace Estimators. 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 $P(X_i = 1|Y = 0)$?
   iii. For all values of i, what is your estimate $P(X_i = 1|Y = 1)$?
   iv. Report your classification accuracy for both MLE and Laplace estimators.

c. For Naïve Bayes trained on `netflix-train.txt` 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 misprediction. Try to explain what went wrong.

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

e. 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 MLE and once with Laplace Estimators.

f. Make sure to turn in a copy of your code for the Naïve Bayes classifier in a zip with all of your code for this problem set.

Continued on the next page...

5. 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 `exp(x)` from the `math` library helpful.

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

   b. Train your algorithm on the data file `netflix-train.txt`. Use learning rate $\eta = 0.0001$ and $3,000$ training steps. Test your algorithm on the data file `netflix-test.txt`, then answer the following questions:

      i. Report your final parameter weights after training.

      ii. Report your classification accuracy.

      iii. According to the weights of your logistic regression function, which five movies are the strongest predictors that a user will like *Love Actually*?

      iv. What is the log-likelihood of the training data when all of the parameters are 0?

      v. What is the log-likelihood of the training data after training?

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

   d. Train your algorithm on the data file `heart-train.txt`. Experiment with using different learning rates η, where you are still testing your algorithm on the data file `heart-test.txt`. 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 η 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, as well as what was the value of the learning rate η you used to obtain it. Explain why you think the learning rate had such an effect on your classification accuracy.

   e. Make sure to turn in a copy of your code for Logistic Regression in a zip with all of your code for this problem set.

6. **[Extra Credit]** For extra credit, try to write a simple neural network algorithm (or even train a Bayesian Network). Train on the data file `netflix-train.txt`. Report the maximum accuracy that you were able to obtain for the data file `netflix-test.txt`.