CS 1571: Homework 4 (programming)

Naive Bayes: Spam Detection (100 pts)

Assigned: November 15, 2017

Due: December 6, 2017

In this assignment, you will create a Naive Bayes classifier for detecting e-mail spam, and you will test your classifier on a publicly available spam dataset using 5-fold cross-validation.

I. Implement Naive Bayes in Python, Java, or C#.

- **Step 1:** Download the Spambase dataset available from the UCI Machine Learning Repository.

  The Spambase data set consists of 4,601 e-mails, of which 1,813 are spam (39.4%). The data set archive contains a processed version of the e-mails wherein 57 real-valued features have been extracted and the spam/non-spam label has been assigned. You should work with this processed version of the data. The data set archive contains a description of the features extracted as well as some simple statistics over those features.

- **Step 2:** Partition the data into 5 folds.

  To estimate the generalization (testing) error of your classifier, you will perform cross-validation. In *k*-fold cross-validation, one would ordinarily partition the data set randomly into *k* groups of roughly equal size and perform *k* experiments (the "folds") wherein a model is *trained* on *k*-1 of the groups and *tested* on the remaining group, where each group is used for testing exactly once. The generalization error of the classifier is estimated by the *average* of the performance across all *k* folds.

  While one should perform cross-validation with random partitions, for consistency and comparability of your results, you should partition the data into 5 groups as follows: Consider the 4,601 data points in the order they appear in the processed data file. ~~Group 1 will consist of points {1-920}, Group 2 will consist of points {921-1840}, Group 3 will consist of points {1841-2760}, Group 4 will consist of points {2761-3680}, Group 5 will consist of points {3681-4601}.~~ **Updated 11/27: Each group will consist of every 5th data point in file order, i.e., Group 1 will consist of points {1,6,11,...}, Group 2 will consist of points {2,7,12,...}, ..., and Group 5 will consist of points {5,10,15,...}.** Finally, Fold *k* will consist of *testing* on Group *k* a model obtained by *training* on the **combined** remaining *k*-1 groups.

- **Step 3:** Create a Naive Bayes classifier by modeling the features in the following way.

  The 57 features are real-valued, and one can model the feature distributions in simple and complex ways. For our assignment, model the features as simple Boolean random variables. Consider a threshold using the overall mean value of the feature (available in the Spambase documentation), and simply compute the fraction of the time that the feature value is above or below the overall mean value for each class. In other words, for feature f_i with overall mean value mu_i, estimate

  - Pr[$f_i <= mu_i$ | spam]
  - Pr[$f_i > mu_i$ | spam]
  - Pr[$f_i <= mu_i$ | non-spam]
  - Pr[$f_i > mu_i$ | non-spam]

  and use these estimated values in your Naive Bayes predictor, as appropriate.

To avoid any issues with zero probabilities, if any of the probability values are 0, simplly replace it with the small value .0014 to avoid multiplying by 0.

II. Evaluate your results.

1. *Error tables*: Create a table with one row per fold showing your false positive, false negative, and overall error rates, and add one final row per table corresponding to the average error rates across all folds. For this problem, the false positive rate is the fraction of non-spam testing examples that are misclassified as spam, the false negative rate is the fraction of spam testing examples that are misclassified as non-spam, and the overall error rate is the fraction of overall examples that are misclassified.

Updated 11/28: See courseweb for detailed instructions regarding What to Submit (code and report)
Updated 11/30: See courseweb for detailed instructions regarding Grading Criteria (to allow you to get partial credit)

What to submit

Submit a .zip package with the name “YourLastName_YourFirstName_HW4.zip”, the package contains the following:

1. **(NaiveBayes.jar, NaiveBayes.exe or NaiveBayes.py)** for your implementation, these files should be runnable from command line. The files **take only one argument which is the Data "spambase.data"** processed version of the dataset. Your code should do the data splitting, training, validation and score reporting. The output of your program should be printed on the console in the following format.

   Line1: Fold_1, false positive, false negative, overall error rates for Fold 1

   Line2: Fold_2, false positive, false negative, overall error rates for Fold 2

   Line3: Fold_3, false positive, false negative, overall error rates for Fold 3

   Line4: Fold_4, false positive, false negative, overall error rates for Fold 4

   Line5: Fold_5, false positive, false negative, overall error rates for Fold 5

   Line6: Avg, avg false positive, avg false negative, avg overall error rates for all folds

2. In case you implemented your code in java or C#, submit also the code **(NaiveBayes.java or NaiveBayes.cs)** alongside the executable files.

3. **A Readme.txt** file, in which mention:
   a. The version of Python you used (if python used), framework used in case of C# or Java.
   b. Any additional libraries needed to run your code.
   c. Any part of your code that is not working.

4. A **Report.pdf or Report.doc** file, in which you report:
   a. **Your scores** in the format specified in the homework document.

   b. Provide **some statistical analysis for the folds** showing the ratio of positive samples and the ratio of negative samples in train data and dev data for each iteration, and how does that affect the scores you get over that iteration.

   c. **Compare** your results with the result of **just choosing the majority class**, provide some analysis of the results you get and summarize your intuition behind the scores you get (**Would it make sense if choosing majority class gets higher results? and if so why do you think that happened?, the same goes for if you got higher scores using naive bayes. Does it make sense and why?)**

Grading Criteria:

Report (30%)

Code (70%)

To allow partial grading in the programming part we will assign different scores for different parts, so that even if your program is corrupted in one of the components you can still have grades for other parts.

<u>So the components we are looking for in the program.</u>

1. Data splitting, your ability to split the data into 5 parts correctly and constructing training and development data out of the 5 parts to run 5 iterations.
2. Calculating the probabilities for each feature using the training data for each iteration.
3. Calculating the probability of each class for each sample in the development data.
4. Finally choosing the max class for each sample.

With these components in mind, the following grades are going to be assigned for each part. And depending on this separation additional intermediate outputs are required.

<u>Score Distribution:</u>

1. Data splitting (20%)
2. Calculating the probabilities for each feature using the training data for each iteration (20%).
3. Calculating the probability of each class for each sample in the development data (20%).
4. Finally choosing the max class for each sample (10%).

<u>Additional outputs:</u>

For us to be able to trace your program and assign partial credits, the following outputs are required.

1. **Data splitting:** After splitting the data, please write simple statistics for each one of the combinations of training and development data. By combinations here we mean the data to be used in each iteration. For example; after splitting the data into 5 parts you are supposed to form the following combinations:
   - Iteration 1, Training data = {part1, part2, part3, part4}, Development data = {part5}
   - Iteration 2, Training data = {part1, part2, part3, part5}, Development data = {part4}
   - Iteration 3, Training data = {part1, part2, part4, part5}, Development data = {part3}
   - Iteration 4, Training data = {part1, part3, part4, part5}, Development data = {part2}
   - Iteration 5, Training data = {part2, part3, part4, part5}, Development data = {part1}

   So, for these combinations create the following table

Iteration	Number of Positive samples in training	Number of Negative samples in training	Number of Positive samples in development	Number of Negative samples in development
1				
2				
3				
4				
5				

<u>This table should be printed on console before the final results and also in your report.</u>

<u>Write your splits in files as well so that we can use it to run on another code and compare with your scores incase yours are not right. So write 2 files or each iteration (Iternumber_Train.txt) and (Iternumber_Dev.txt) these two files should be in the same format as the original data.</u>

2. **Calculating the probabilities** for each feature using the training data for each iteration

In each iteration you should calculate the different probabilities from your data, you should end up with four different probabilities for each feature.

- Pr[fi <= mui | spam]
- Pr[fi > mui | spam]
- Pr[fi <= mui | non-spam]
- Pr[fi > mui | non-spam]

So using these probabilities construct the following table in **<u>your report. Also submit the same data in an excel sheet</u>** to ease importing it, so we can calculate step 3 and 4 using the probabilities you calculated and compare it with your final scores.

iter	Pr(F1<=mui | spam)	Pr(F1>mui | spam)	Pr(F1<=mui | non-spam)	Pr(F1>mui | non-spam)	Pr(F2<=mui | spam)	Pr(F2>mui | spam)	etc….
1							
2							
3							
4							
5							

3&4 – For these two points the grade will be determined based on the final scores you calculated and the scores we calculated using your probabilities.

<u>What to submit (updated):</u>

- Your code as specified by the original what to submit file document.

- The readme file as specified by the original what to submit file document.
- The report as specified by the original what to submit file document with the additional information we mentioned here (in case you want to be considered for partial crediting)
- Excel sheet for probabilities (in case you want to be considered for partial crediting)
- Data splits (Iternumber_Train.txt) & (Iternumber_Dev.txt) for each iteration (in case you want to be considered for partial crediting)

Final Notes:

- If you failed to provide outputs in the specified formats and your final scores are not coherent with the expected scores, we have no way of giving you partial credit and you may lose the whole program grade.

- If no report is attached or your report is not informative, you may lose all the report points.