Assignment 3: Naive Bayes Classifier and Text Classification

UVA CS 4501 - 001 / 6501 - 007 :
Introduction to Machine Learning and Data Mining (Fall 2014)

Out: Oct. 16th, 2014
Due: Oct. 26th 2014, **Sunday, midnight**,

a *The writing portion of the assignment should be submitted in the PDF format through Collab. Latex based PDF is recommended.*

b *This coding portion of the assignment should be submitted in the format of source code through Collab.*

c *This assignment has one programming task to improve your understanding of naive bayes classification and its application on text categorization.*

d *For questions and clarifications, please post on piazza. TA Nick (ncj2ey@virginia.edu) or Beilun (bw4mw@virginia.edu) will try to answer there.*

e *Policy on collaboration:*

Homework will be done individually: each student must hand in their own answers. It is acceptable, however, for students to collaborate in figuring out answers and helping each other solve the problems. We will be assuming that, with the honor code, you will be taking the responsibility to make sure you personally understand the solution to any work arising from such collaboration.

e *Policy on late homework: Homework is worth full credit at the beginning of class on the due date. Each student has three extension days to be used at his or her own discretion throughout the entire course. Your grades would be discounted by 15% per day when you use these 3 late days. You could use the 3 days in whatever combination you like. For example, all 3 days on 1 assignment (for a maximum grade of 55%) or 1 each day over 3 assignments (for a maximum grade of 85% on each). After you've used all 3 days, you cannot get credit for anything turned in late.*

1 Naive Bayes Classifier (TA: Beilun)

In this programming assignment, you are expected to implement the **Naive Bayes Classifier** using python for a text-based movie review classification task. A ZIP file "data_sets_naive_bayes.zip" including two sets of data samples (i.e. training set and test set) for movie reviews is provided to you through collab attachment. Please following the required file and function naming.

We expect you to submit a source-code file named as naiveBayes.py containing the necessary and required functions for training, testing and evaluations.

For a naive Bayes classifier, when given an unlabeled document d, the predicted class

$$c_d^* = \underset{c}{\operatorname{argmax}}\, \mathbb{P}(c|d)$$

, where c is the value of the target class variable. For the target movie review classification task, $c =$ **pos** or **neg**. For example, if $\mathbb{P}(c = pos|d) = \frac{3}{4}$ and $\mathbb{P}(c = neg|d) = \frac{1}{4}$, we use the MAP rule to classify the document d into the "postive" class. The conditional probability $\mathbb{P}(c|d)$ is calculated through **Bayes' rule**,

$$\mathbb{P}(c|d) = \frac{\mathbb{P}(c)\mathbb{P}(d|c)}{\mathbb{P}(d)} \propto \mathbb{P}(c)\mathbb{P}(d|c)$$

This assignment requires you to implement three types of naive bayes classification, wherein the first two follow the Multinomial assumption and the third is under the Bernoulli assumption.

1.1 Preprocessing

The rough result of the preprocessing is illustrated in Lecture-15's slide-20.

- (0) Normally, as the first step, you will build a vocabulary of unique words from the training corpus, being ranked with their frequency. Then you will just use the top K words that appearing more than a certain times in the training corpus.
- (Q1) For this homework, to get a consistent result from all the students, please use the following predefined dictionary with words: {love, wonderful, best, great, superb, still, beautiful, bad, worst, stupid, waste, boring, ?, ! }. Besides, for the token "love", you should also consider the tokens "loving", "loved", "loves" since their stemmed version will be the same as token "love".
- We could add one entry "UNKNOWN" as the first word in your dictionary to handle all those words not included in the current dictionary.

1.2 Build "bag of words" (BOW) Document Representation

- The rough process of building "bag of words" is illustrated in Lecture-14's slide-35 and slide-36.
- (Q2) You are required to provide the following function to convert a text document into a feature vector:
  BOWDj = naiveBayesMulFeature.transfer(fileDj, vocabulary)
- (Q3) Read in the training and test documents into BOW vector representations using the above function. Then store features into matrix Xtrain and Xtest, and use ytrain and ytest to store the labels. You are required to provide the following function to convert a text document into a feature vector:
  Xtrain, Xtest, ytrain, ytest = naiveBayesMulFeature.loadData(textDataSetsDirectoryFullPath)
- Here: "textDataSetsDirectoryFullPath" is the real full path of the file directory that you get from unzipping the datafile. For instance, it is "/HW3/data_sets/" on the instructor's laptop.

1.3 Multinomial Naive Bayes Classifier (MNBC) Training Step

- The rough process of training / estimating parameters for MNBC is described in Lecture-15's slide-39 or Lecture-14's slide-51+52.
- We need to learn the $\mathbb{P}(c_j)$ and $\mathbb{P}(w_i|c_j)$ through the training set. Through MLE, we use the relative-frequency estimation with Laplace smoothing to estimate these parameters. (L15-slide-39)
- Since we have the same number of positive samples and negative samples, $\mathbb{P}(c = -1) = \mathbb{P}(c = 1) = \frac{1}{2}$.
- (Q4) You are required to provide the following function (and module) for grading:
  thetaPos, thetaNeg = naiveBayesMulFeature.train(Xtrain, ytrain)
- (Q5) Provide the resulting value of thetaPos and thetaNeg into the writing.

Note: Pay attention to the MLE estimator plus smoothing (L15-slide-39); Here we choose $\alpha = 1$.

1.4 Multinomial Naive Bayes Classifier (MNBC) Testing+Evaluate Step

- (Q6) You are required to provide the following function (and module) for grading:
  yPredict, Accuracy= naiveBayesMulFeature.test(Xtest, ytest)
  Add the resulting Accuracy into the writing.
- (Q7) Use "sklearǹnaive_bayes.MultinomialNB" from the scikit learn package to perform training and testing. Compare the results with your MNBC. Add the resulting Accuracy into the writing.

Important: Do not forget perform **log** in the classification process.(Lecture 14 slides 55)

1.5 Multinomial Naive Bayes Classifier (MNBC) Testing through non-BOW feature representation

- The rough pipeline is illustrated in Lecture-15's slide-20 and slide-25. For the step of classifying a test sample using MNBC, It is actually not necessary to first perform the BOW transformation for feature vectors.
- (Q8) You are required to provide the following function (and module) for grading:
  yPredictOne= naiveBayesMulFeature.testDirectOne(XtestTextFileNameInFullPathOne)
- (Q9) Use the above function on all the possible testing text files, calculate the "classification accuracy" based on "yPredict" versus the testing label. Please add the resulting accuracy into the writing. You are required to provide the following function (and module) for grading:
  yPredict, Accuracy= naiveBayesMulFeature.testDirect(testFileDirectoryFullPath)
- Here: "testFileDirectoryFullPath" is the real full path of the directory with the test set that you get from unzipping the datafile. For instance, it is "/HW3/data_sets/test_set/" on the instructor's laptop.

1.6 Multivariate Bernoulli Naive Bayes Classifier (BNBC)

- The rough process of training / estimating parameters for BNBC is described in Lecture-14's slide-15,16,40,51.
- We need to learn the $\mathbb{P}(c_j)$, $\mathbb{P}(w_i = false|c_j)$ and $\mathbb{P}(w_i = true|c_j)$ through the training. MLE gives the relative-frequency as the estimation of parameters. We will add with Laplace smoothing for estimating these parameters. (Lecture-14's slide-15,16,40,51.)
- Essentially, we simply just do counting to estimate $\mathbb{P}(w_i = true|c)$.

$$\mathbb{P}(w_i = true|c) = \frac{\#\text{files which include } w_i \text{ and are in class c} + 1}{\#\text{files are in class c} + 2}$$

$$\mathbb{P}(w_i = false|c) = 1 - \mathbb{P}(w_i = true|c)$$

- Since we have the same number of positive samples and negative samples, $\mathbb{P}(c = -1) = \mathbb{P}(c = 1) = \frac{1}{2}$.
- (Q10) You are required to provide the following function (and module) for grading:
  thetaPosTrue, thetaNegTrue= naiveBayesBernFeature.train(Xtrain, ytrain)
- (Q11) Provide the resulting parameter estimations into the writing.
- (Q12) You are required to provide the following function (and module) for grading:
  yPredict, Accuracy= naiveBayesBernFeature.test(Xtest, ytest)
  Add the resulting Accuracy into the writing.

1.7 How will your code be checked ?

The following functions will be run in order to check your toolset. "textDataSetsDirectoryFullPath" and "testFileDirectoryFullPath" are string inputs.

```
Xtrain, Xtest, ytrain, ytest = naiveBayesMulFeature.loadData(textDataSetsDirectoryFullPath)
thetaPos, thetaNeg = naiveBayesMulFeature.train(Xtrain, ytrain)
yPredict, Accuracy= naiveBayesMulFeature.test(Xtest, ytest)
yPredict, Accuracy= naiveBayesMulFeature.testDirect(testFileDirectoryFullPath)
thetaPosTrue, thetaNegTrue= naiveBayesBernFeature.train(Xtrain, ytrain)
yPredict, Accuracy= naiveBayesBernFeature.test(Xtest, ytest)
```

Congratulations ! You have implemented a state-of-the-art machine-learning toolset for an important web data mining task !