Assignment 5: Naive Bayes Classifier for Text Classification

UVA CS 4501004 / 6316 :
Machine Learning (Fall 2016)

Out: Oct. 31, 2016
Due: Nov. 12 at 11:55pm, 2016 in the course Collab

a *The assignment should be submitted in the PDF format through Collab. If you prefer hand-writing the writing part of answers, please convert them (e.g., by scanning) into PDF form.*

b *For questions and clarifications, please post on piazza. TA Muthu (mc4xf@virginia.edu) or Jack (jjl5sw@Virginia.EDU) will try to answer there.*

c *Policy on collaboration:*

Homework should 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.

d *Policy on late homework: Homework is worth full credit at the midnight 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 for Text-base Movie Review Classification (TA: Muthu and Jack)

In this programming assignment, you are expected to implement **different Naive Bayes Classifiers** 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 attachments. Please follow the following naming rules wrt files and functions.

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 "positive" 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 classifiers, among which the first two follow the Multinomial assumption and the third uses the multivariate Bernoulli assumption.

1.1 Preprocessing

(Q1) You are required to implement the first choice of preprocessing described as following.

You can get 1 point of extra credit if you also implement the second choice of preprocessing and discuss the classification results based on it.

- (First choice): to get a consistent result from all the students, please use a predefined dictionary including the following words: {love, wonderful, best, great, superb, still, beautiful, bad, worst, stupid, waste, boring, ?, !, UNK }.
  - Besides, for the token "love", you should also consider the tokens "loving", "loved", "loves" since their stemmed version will be the same as the token "love". In other words, the frequency of love should include the words "love, loving, loves, loved". In order to do that, you don't need to use the NLTK. You can do a simple string.replace("loved", "love"), string.replace("loves", "love"), string.replace("loving", "love"). No other preprocessing is necessary (e.g. stemming, stopword removal).
  - UNK represents unknown words not included in the dictionary.
  - In summary, thetaPos and thetaNeg are each vectors of length 15.
- (Second choice): 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 (e.g. 3 times) in the whole training corpus.
  - It is recommended that you use stemming and stopword removal for this (you can use a python package such as NLTK).

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

- (Q2) You are required to provide the following function to convert a text document into a feature vector:

  $BOWDj = transfer(fileDj, vocabulary)$

  where fileDj is the location of file j
- (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:

  $Xtrain, Xtest, ytrain, ytest = loadData(textDataSetsDirectoryFullPath)$
  - "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.
  - loadData should call transfer()

  Note: Xtrain and Xtest are matrices with each column representing a document (in BOW vector format). ytrain and ytest are vectors with a label at each position. These should all be represented using a python list or numpy matrix.

1.3 Multinomial Naive Bayes Classifier (MNBC) Training Step

- 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.
- 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 writeup.

  Note: Pay attention to the MLE estimator plus smoothing; Here we choose $\alpha = 1$.

  Note: thetaPos and thetaNeg should be python lists or numpy arrays (both 1-d vectors)

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

- (Q6) You are required to provide the following function (and module) for grading:

  $yPredict, Accuracy = naiveBayesMulFeature\_test(Xtest, ytest, thetaPos, thetaNeg)$

  Add the resulting Accuracy into the writeup.

- (Q7) Use "sklearn.naive_bayes.MultinomialNB" from the scikit learn package to perform training and testing. Compare the results with your MNBC. Add the resulting Accuracy into the writeup.

  Important: Do not forget perform **log** in the classification process.

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

- 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:

  $yPredictOne = naiveBayesMulFeature\_testDirectOne(XtestTextFileNameInFullPathOne, thetaPos, thetaNeg)$

- (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:

  $yPredict, Accuracy = naiveBayesMulFeature\_testDirect(testFileDirectoryFullPath, thetaPos, thetaNeg)$

  Note: "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)

- 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.

- 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:

  $yPredict, Accuracy = naiveBayesBernFeature\_test(Xtest, ytest, thetaPosTrue, thetaNegTrue)$

  Add the resulting Accuracy into the writing.

1.7 How will your code be checked ?

In collab, you will find the sample codes named "naiveBayes.py". "textDataSetsDirectoryFullPath" and "testFileDirectoryFullPath" are string inputs. We will run the command line: "python naiveBayes.py textDataSetsDirectoryFullPath testFileDirectoryFullPath" to check your code if it can print the result of the following functions in the table.

thetaPos, thetaNeg = naiveBayesMulFeature_train(Xtrain, ytrain)
yPredict, Accuracy= naiveBayesMulFeature_test(Xtest, ytest, thetaPos, thetaNeg)
yPredict, Accuracy= naiveBayesMulFeature_testDirect(testFileDirectoryFullPath, thetaPos, thetaNeg, vocabulary)
thetaPosTrue, thetaNegTrue= naiveBayesBernFeature_train(Xtrain, ytrain)
yPredict, Accuracy= naiveBayesBernFeature_test(Xtest, ytest,thetaPosTrue, thetaNegTrue)

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