Assignment 3: Naive Bayes Classifier for Text Classification and Multiple classifiers for Image-based OCR

UVA CS 6316 :
Machine Learning (Fall 2015)

Out: Oct. 13, 2015
Due: Nov. 12 midnight 11:55pm, 2015 @ 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 Ritas (rs3zz@virginia.edu) or Beilun (bw4mw@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: 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-13 slide-28 to 31.

- (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 = 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 = 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-14's slide-11 to 19 and slides 31 to 35.
- 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. (Lecture-14-slide-34)
- 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 (L14-slide-34); 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, thetaPos, thetaNeg)
  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 20)

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

- The rough pipeline is illustrated in Lecture-14's slide-12 to slide-18. 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)
- 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-25 to slide-31.
- 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-13's slide-9 to 13 and L14-slide-34)
- 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)
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 data mining task !

2 Content-based image classification through Supervised Classifiers in Python (TA: Ritas)

An online tutorial might be helpful to you for finishing this assignment,
`http://blog.yhathq.com/posts/image-classification-in-Python.html`

For this assingment, you will be comparing different ML techniques for recognizing hand-written digits. The data set is available as zip.train.gz and zip.test.gz. Each example in the train and test data is a the number in question followed by 256 grayscale values representing a 16×16 image. Values have already been normalized. You can read a full description of the data set in the zip.info.txt file available in collab.

For each question, you will apply a different model to the dataset using scikit-learn. The purpose of this assignment is to further develop your intuition in regards to machine learning methods and to give you experience with dimensionality reduction. Visualization of some samples are provided in the subdir "image_digits" in the attached data ZIP file.

2.1 Decision Tree

Using sklearn.tree.DecisionTreeClassifier, apply the decision tree classifier to the dataset and include the following within the written report. Make sure to fill in the method stub for number_recognition.decision_tree(train, test).

- Choose three parameters to customize for the decision tree, and run your model using those parameters.
- What is the testing error using those parameters?
- What should each of those parameters do for the model?

2.2 K Nearest Neighbors

Use scikit-learn's sklearn.neighbors.KNeighborsClassifier for these questions and fill in the respective function in the provided file.

- Select five different values for k and report the test and training error.
- Based on this information, what value of k would you choose for this data set and why?
- What explanation do you have for KNN's performance on this data set? How do you think distance weighting would impact performance, why?

2.3 Neural Net - Perceptron

Scikit-learn offers a fundamental neurel network (basic module) known as a perceptron.

It is available as sklearn.linear_model.Perceptron. The neural_net function should be filled in for this question. It is a linear model that depends on the data's linear seperability.

- Report the perceptron's test error for this data set.
- What do you think explains this performance?

2.4 Support Vector Machine

You should be well acquainted with the SVM classifier after homework two. We will use the same library (sklearn.svm.SVC) for this homework problem.

- How do the rbf, linar, and polynomial kernels compare on this data set? Please include the parameters you use for each one.
- Theoretically, do you think a SVM is well suited to this problem, why or why not?

2.5 PCA

(optional) Principal component analysis allows us to reduce the dimensions of our data while retaining as much variance as possible, thus retaining information to separate our data points. Scikit learn provides multiples ways to perform dimension reduciton with PCA. For this problem, sklearn.decomposition.RandomizedPCA is recommended.

- Using your best K value from problem 2, select three different values for PCA and present your test error results. Why would you want to use PCA with KNN?
- Do the same for your implementation of the SVM. What do you notice about its impact on SVM's performance and how do you explain it?

2.6 Submission

Please submit your source code and a pdf containing your written answers via collab.

2.7 Grading

In collab, you will find starter code in number_recognition.py. This file will be run from the command line. For example:

python dtree path/to/training_data path/to/testing_data

Feel free to add any methods or classes to your implementation, so long as your preserve the given command line behaviour for your submission.

Congratulations ! You have implemented a state-of-the-art machine-learning toolset for an important OCR image labeling task !

You can do a lot of cool things by extending the above toolset, for instance, build an app recognizing handwritten digits using "node.js", e.g.
`http://blog.yhathq.com/posts/digit-recognition-with-node-and-python.html`

Figure 1: Description of this data set (zip.info.txt file)

```
Normalized handwritten digits, automatically
scanned from envelopes by the U.S. Postal Service. The original
scanned digits are binary and of different sizes and orientations; the
images  here have been deslanted and size normalized, resulting
in 16 x 16 grayscale images (Le Cun et al., 1990).

The data are in two gzipped files, and each line consists of the digit
id (0-9) followed by the 256 grayscale values.

There are 7291 training observations and 2007 test observations,
distributed as follows:
         0    1   2   3   4   5   6   7   8   9 Total
Train 1194 1005 731 658 652 556 664 645 542 644 7291
 Test  359  264 198 166 200 160 170 147 166 177 2007

or as proportions:
         0    1   2    3    4    5    6    7    8    9
Train 0.16 0.14 0.1 0.09 0.09 0.08 0.09 0.09 0.07 0.09
 Test 0.18 0.13 0.1 0.08 0.10 0.08 0.08 0.07 0.08 0.09


Alternatively, the training data are available as separate files per
digit (and hence without the digit identifier in each row)

The test set is notoriously "difficult", and a 2.5% error rate is
excellent. These data were kindly made available by the neural network
group at AT&T research labs (thanks to Yann Le Cunn).
```