Comp 135 Intro ML: Project 1

Tufts 2018 Spring, Instructor: Liping Liu, credit to our TA team

Due date: Wednesday, 02/28

1 Introduction

In this assignment you will implement the Naive Bayes algorithm with smoothing for the task of sentiment classification, which helps to identify positive and negative product reviews.

2 Text Data for Sentiment Analysis

We have three datasets collected from three domains: imdb.com, amazon.com, yelp.com. Each dataset consists of sentences with sentiment labels (1 for positive and 0 for negative) extracted from . These form 3 datasets for the assignment.

Each dataset is given in a single text file, with each line as an instance. Each line is a list of space separated words, which essentially a sentence, followed by a tab character, and then followed by the label. Here is a snippet from the yelp dataset:

Crust is not good. *0*
Best burger ever had! *1*

You are suggested to remove all the punctuation, numeric values, and convert upper case to lower case for each example so that the same word will treated in the same way in the data.

3 Implementation

3.1 Naive Bayes for Text Categorization

In this assignment you are ask to implement a Naive Bayes classifier, which has been discussed in class, to predict sentiment labels of sentences.

3.2 Document to a vector – bag-of-words

We can convert a document to a vector by checking whether a word appears in the document. Each word corresponds to a feature. If a word j appear in the document, then the j-th feature of the document is 1; otherwise its j-th feature is 0. In this way, each document is converted to a vector with length of the vocabulary size.

Consider the illustration from class with positive examples: **what a nice day** and **a green cat chased a green dog** and negative example **green umbrella a nice day**. Then the feature table is given by

what	a	nice	day	green	cat	chased	dog	umbrella	Class
1	1	1	1	0	0	0	0	0	+
0	1	0	0	1	1	1	1	0	+
0	1	1	1	1	0	0	0	1	-

3.3 Training and testing

Your learning algorithm will be evaluated by cross-validation, therefore, you want to evaluate your classifier before you submit it. Please make clear how you have evaluated your classifier in the report. Your reported performance values should be obtained with the same evaluation method.

3.4 Smoothing

When you estimate the probability $p(x_{.j} = 1|y = y_0)$, which is the probability that word j will be present if the class is given as y_0. The estimation with smoothing should be $\frac{\#[x_{.j}=1 \wedge y=y_0]+m}{\#[y=y0]+md}$, with d being the vocabulary size. The variable m is the smoothing factor. You are asked to try different m values to see its effects on performance (see below).

3.5 Practical issues

When you work with probabilities, try to work in the logarithm space (or logarithm of probabilities) to prevent underflow problems.

If the vocabulary size is very large, you can skip some words in the data. You may want to skip words that are very frequent or very infrequent (why?).

4 Experiments

Once all the above is implemented please run the following tests and report the results. You are requested to run Naive Bayes many times on combinations of datasets, and parameters. With a good implementation the overall time should not be high. But please plan ahead to make sure you can complete the assignment on time.

- Run Naive Bayes with smoothing parameter m = 0.1, 0.5, 2.5, 12.5. Plot the cross validation performance as a function of the smoothing parameter. What observations can you make about the results?

- Choose a small vocabulary (size d) of words and check whether how the performance changes. Try different d values, 1000, 500, 250, 125, and report what performance you get for these different sizes of vocabulary.

- (Optional) Separate a dataset of N instances into three subsets, training, validation, and test subsets, to imitate real applications. Fix the size of test subset to 100 instances, and split the rest of the data into training and validation subsets. The test subset is like the real data – error rate on test data is something you really care (just like errors in real test data cost real loss). Now you want to estimate this error rate without touching the test subset. So you train a model on training subset and use validation subset to make the estimation. You are asked to try different split ratio of training and validation subsets. Let $N' = N - 100$ be

the size of the data you can access. For example, you are asked to try to use [0.1N', 0.5N', 0.8N', 0.95N'] instances as your training set. Plot the validation error rate against the ratio of training set. Also plot the the real test error rate. Report which training ratio gives you the best estimation of test error rate.

5 Interface of the implementation

You are supposed to implement the classifier with functions using the following interfaces in Python. If you are using the Matlab, please use the same interfaces.

Let N_{train} as the number of examples in the training data, N_{test} is the number of examples in the testing data, and d is the number of features (vocabulary size).

```
import numpy
import scipy

def train(X_train, y_train, train_opt):
    # Train the model with X_train, y_train and train_opt.
    # Inputs:
    #   1) X_train: [N_train*d] numpy array, the training features.
    #   2) y_train: [N_train] numpy vector, the training labels.
    #   3) train_opt: [dict] taking form {"smooth_param": 0.1}.
    #                 The model is trained with the smooth parameter.
    # Output:
    #   1) trained_model: [dict], contains the trained model.
    return trained_model

def test(X_test, trained_model):
    # Predict the labels for X_test, given the trained model.
    # Input:
    #   1) X_test: [N_test*d] numpy array, the testing features.
    #   2) trained_model: [dict], contains the trained model.
    # Output:
    #   1) y_pred: [N_test] numpy vector, the predicted labels.
    reutrn y_pred

def evaluate(y_test, y_pred):
    # Evaluate between y and y_pre, using the options in eval_opt.
    # Input:
    #   1) y_test: [N_test] numpy vector, the testing labels.
    #   2) y_pred: [N_test] numpy vector, the predicted labels.
    # Output:
    #   1) error_rate: a float number in [0, 1],
    #                  the error rate of the prediction.
    return error_rate

def main():
```

```
# You should implement your own main function. Here is just an
# reference

# use what ever preprocessing package you like to get data
# matrix from data file. One example:
X, Y = read_in_data_function(datafile_name, vocabulary)

# split data into two folds
X_train, Y_train, X_test, Y_test = split_data(X, Y, train_test_ratio)

# if you do the experiment of training, validating, and testing
# further split the training set into training and validation
# subsets
# X_validation, Y_validation, X_test, Y_test =
#                 split_data(X_train, Y_train, train_valid_ratio)


trained_model = train(X_train, Y_train, train_opt)
Y_pred = test(X_test, trained_model)
error_rate = evaluate(Y_test, Y_pred)
print('The error_rate of my classifier is ' + str(error_rate))
```

6 Programming Language and use of Libraries/Modules

You may write your program in any language as long as your code runs on our server homework.eecs.tufts.edu. You are of course allowed to use basic I/O, math library, randomization and other basic language facilities. But you should write your own code for all the portions described above. For example, you are free to use *numpy* or *scipy* libraries in Python, but you cannot use *scikit-learn*.

7 Submitting your assignment

7.1 What to submit

You should submit the following items electronically:

- All your code for data processing, learning algorithms, experiments, test programs, etc. Please write clear code and document it as needed.
  Please make sure that your code runs on homework.eecs.tufts.edu. Please include a README file with instructions how to compile and run your code to reproduce the results of experiments. If this is nontrivial, please include a script to compile and run your code.
  Your submitted code should assume that the data files are in the same directory as the program. There is no need to submit the original data files.

- A short report with the results and plots as requested and a discussion with your observations

from these plots. Please make sure that your discussion responds to questions posed in the assignment text.

8 When and How to Submit

Please submit electronically using provide by 4:30pm of the due date. Put all the files mentioned in the previous subsection into a *.zip* file. For example call it myfile.zip. Then submit using provide comp135 pp1 myfile.zip.

8.1 Grading

Your assignment will be graded based on the code, its clarity, documentation and correctness, the presentation of the results/plots, and their discussion.