REPORT: CSE5243 Homework #3

Xin Jin@CSE, OSU
jin.967@osu.edu

October 26, 2018

Introduction

This assignment is the second part of a longer-term project. The objective is to give you the experience of building classifiers for your preprocessed real data in HW #2.

1 Preprocessing

In the HW #3 assignment, I built and evaluated several classifiers to predict the sentiment of a given sentence. And I completed this preprocessing procedure by the following steps.

1.1 Dataset split

In this part, I first load the dataset from result of the HW #2 project. I obtain the dataFrame, in which the feature is the word and its one-hot representation. The original feature vector is **FV-1.** Then I use the sklearn package to split the data set into training set, test set, validation set and the set size is 60%, 20% and 20%.

1.2 Feature selection

In this part, I first exclude the stop words out of the origianl feature vector. Then I use the TF-IDF value to extract the key **1000** feature words vector as **FV-2**. Then I use the key 1000 feature to prune the training set, test set and validation set.

2 Training Classifiers

To get the suitable parameters, I use the training set and validation set to train the algorithms to get the ideal parameters for the classifiers.

2.1 K-nearest neighbors

For the KNN classifier, I designed my own classifier and I also used the knn classifier from the *sklearn* package.

2.1.1 My own version

In my own knn classifier, I first calculate the distance between validation examples and training set to get the k neighbors and then vote the neighbors to get the label classification for the validation example. Then the classification is done.

To get the optimal K value, I tried the list as the K value in the list [2,5,8,...,62] by 30 iterations. Then I compared the accuracy and running time to figure out which k value is better

After training, I get K=8 as the optimal K value for FV-1 and FV-2. And the training command is as follows (the runing time is within 4000 seconds, the complex results are not shown here):

```
Command Line
$ python classification.py train knn1
```

2.1.2 KNN from sklearn package

By using the knn classifier from the sklearn package, it is very easy to train the classifier. The sklearn.neighbors class supply the KNeighborsClassifier() function for users. And I only need to train the k value to get the optimal classifier.

To get the optimal K value, I tried the the K value in the list [2,5,8,...,62] by 30 iterations. Then I compared the accuracy and running time to figure out which k value is better.

After training, I get K=11 as the optimal K value for FV-1 and FV-2. And the training command is as follows (the runing time is within 300 seconds, the complex results are not shown here):

```
Command Line
$ python classification.py train knn2
```

2.2 Naive Bayes

For Naive bayes classifier, I use the sklearn package to get the functions.

There are three kinds of core functions, *GaussianNB, MultinomialNB, BernoulliNB*, in the sklearn.naive_bayes class. I train the classifiers by using the GaussianNB, MultinomialNB, BernoulliNB functions one by one in 10 rounds. Then I compared the average accuracy and running time to figure out which core function is better within 10 rounds.

After training, I use the *MultinomialNB* as the core function in my classifier. And the training command is as follows (the runing time is within 100 seconds, the complex results are not shown here):

```
Command Line
$ python classification.py train nb
```

2.3 Logistic regression

For logistic regression, I use the sklearn package to get the functions.

There are four kinds of core slovers, *'newton-cg', 'lbfgs', 'sag', 'saga'*, in the sklearn.linear_model class and LogisticRegression() function. I train the classifiers by using the 'newton-cg', 'lbfgs', 'sag', 'saga' solvers one by one in 10 rounds. Then I compared the average accuracy and running time to figure out which core solver is better within 10 rounds.

After training, I use the *'lbfgs'* as the core solver in my classifier. And the training command is as follows (the runing time is within 300 seconds, the complex results are not shown here):

```
Command Line
$ python classification.py train lr
```

2.4 Support vector machine

For support vector machine, I use the sklearn package to get the functions.

There are three kinds of core functions, *SVC, LinearSVC, NuSVC*, in the sklearn.svm class. I train the classifiers by using the SVC, LinearSVC, NuSVC functions one by one in 10 rounds. Then I compared the average accuracy and running time to figure out which core function is better within 10 rounds.

After training, I use the *GaussianNB* as the core function in my classifier. And the training command is as follows (the runing time is smaller than 500 seconds, the complex results are not shown here):

Command Line

```
$ python classification.py train svm
```

3 Classification

Afther the training procedure, I have the classifiers with the optimal parameters. So I can online classify and predict the sentiment labels for the test instances in test set. For the different classifiers, the command is as follows:

3.1 KNN (my own version)

The training command and results are as follows (the runing time is within 100 seconds):

Command Line

```
$ python classification.py classify knn1

*** Classification by KNN (my own KNN algorithm)***
For K-nearest neighbors classifier: the accuracy with FV-1 is 0.6133,
the accuracy with FV-2 is 0.6683.
For K-nearest neighbors classifier: the running time with FV-1 is 63.3246,
the running time with FV-2 is 25.6959.
```

3.2 KNN(from the sklearn package)

The training command and results are as follows (the runing time is within 10 seconds):

Command Line

```
$ python classification.py classify knn2

*** Classification by KNN (KNN algorithm from sklearn package) ***
For K-nearest neighbors classifier: the accuracy with FV-1 is 0.6850,
the accuracy with FV-2 is 0.6950.
For K-nearest neighbors classifier, the running time with FV-1 is 6.4026,
the running time with FV-2 is 1.5406.
```

3.3 Naive Bayes

The training command and results are as follows (the runing time is within 1 second):

Command Line

```
$ python classification.py classify nb

*** Classification by Naive Bayes from sklearn package***
For naive bayes classifier: the accuracy with FV-1 is 0.820000,
the accuracy with FV-2 is 0.783333
For naive bayes classifier: the running time with FV-1 is 0.2665,
the running time with FV-2 is 0.0367.
```

3.4 Logistic regression

The training command and results are as follows (the runing time is within 2 seconds):

Command Line

```
$ python classification.py classify lr

*** Classification by logistic regression from sklearn package***
For logistic regression classifier: the accuracy with FV-1 is 0.798333,
the accuracy with FV-2 is 0.778333
For logistic regression classifier: the running time with FV-1 is 1.0384,
the running time with FV-2 is 0.1783.
```

3.5 Support vector machine

The training command and results are as follows (the runing time is within 20 seconds):

Command Line

```
$ python classification.py classify svm

*** Classification by support vector machine from sklearn package***
For support vector machine classifier, the accuracy for test set with FV-1 is
 0.826667, the accuracy for test set with FV-2 is 0.781667
For support vector machine classifier: the running time with FV-1 is
15.3823, the running time with FV-2 is 3.9340.
```

4 Results and Analysis

After training and classification, I collect all the results and try to analyze them here.

4.1 Offline training time

For offline training, I only ran the training code for one time, because the rounds in the training procedure is big enough. Here, I list all the offline training time in the following Table 1 to give a comparison:

From the table, we can see that the runing time of my KNN classifier is the biggest one, which means it is the most unefficient one. The knn algorithm from the sklearn package is much more efficient than mine. There are many places of my code that I can improve. And the naive bayes classfier is the most quick one to be trained.

By comparing the running time of FV-1 and FV-2, we can see feature pruning saves me plenty of time than unpruning!

Table 1: Offline training time for different classifiers

	KNN(my version)	KNN(from package)	naive bayes	logistic regression	SVM
FV-1 (seconds)	3334.895	137.347	5.308	194.292	349.460
FV-2 (seconds)	505.929	33.914	1.155	40.780	88.665

4.2 Online classification time

For online training, I ran the training code for several times, then I calculate the average classification time for each classifer . Here, I list all the online classification time in the following Table 2 to give a comparison:

Table 2: Online classification time for different classifiers

	KNN(my version)	KNN(from package)	naive bayes	logistic regression	SVM
FV-1 (seconds)	65.111	6.456	0.367	1.138	16.392
FV-2 (seconds)	23.034	1.341	0.038	0.188	4.334

From the table, we can see that the runing time of my KNN classifier for classification is the biggest one, which means it is the most unefficient one. Compared with the KNN classification algorithm from the sklearn package, my own knn algorithm is much slower. It means that my code has many unefficient parts, such as two many loops. And the naive bayes classfier is the most quick one to be trained.

By comparing the running time of FV-1 and FV-2, we can see feature pruning saves me plenty of time than unpruning!

4.3 Online classification accuracy

Like the online classificationpart, I ran several time for the classification and get the average accuracy for FV-1 and FV-2. The result is in Table 3 and analysis are as follows:

Table 3: Online classification accuracy for different classifiers

	KNN(my version)	KNN(from package)	naive bayes	logistic regression	SVM
FV-1 (seconds)	0.681	0.616	0.825	0.800	0.841
FV-2 (seconds)	0.688	0.660	0.783	0.791	0.798

From the table we can see that my KNN classifier have better perfomance than knn algorthms from sklearn package both in FV-1 and FV-2. But these two kinds of knn classifiers both have the less performance than naive bayes, logistic regression and support vector machine. And the support vector machine classifier have the best performance among all the classifiers.

When considering the difference between the accuracy of FV-1 and that of FV-2, we can see pruning can help improve the performance of the two kinds of knn classifiers. But for naive bayes, logistic regression and support vector machine classifiers, pruning doesn't help to improve the performance of the classifiers.

So we can draw a conclusion that the pruning will help save the offline training time and online classification time of classifiers, but it doesn't necessarily help on improving the performace of the accuracy.

5 Look ahead

5.1 Something I want to do but didn't

In this project, we use the one-hot word representation method to represent the word. It is very simple and straightforward. But it is not very efficient.

I planed to use the word2vec word embedding method to represent the words but I didn't have that time to finish it. One advantage of word2vec representation is that we can get the relationship of the words in different position in the sentence. This is a very important information in natural language processing.

I also planed to use nerual netwok to train the classifier, especially for the DNN. I think if I can use the DNN or RNN, the classification accuracy will be higher than 90%.

5.2 Future Plan

For the next project, I will try to use the word2vec and glove to represent the words. Then I will try to use the nerual network, like CNN, DNN, RNN and LSTM, as the classifiers.

6 Source and reference

For this project, I have to mention the following acknowledgement:

1. Project created using packages: numpy, sklearn, pandas, re, operator, nltk, sys, time.
2. Code developed using the Pycharm IDE and python 3.6.
3. Project created citing from sklearn as reference for code (was changed by me during implementation to be my own unique code).
4. Started to follow this guide, but I changed my mind near the beginning and figured out most of it on my own. Code was changed by me during implementation anyways, even when I was following it.
5. The dataset was originally used in the following paper: From Group to Individual Labels using Deep Features", Kotzias et al., SIGKDD 2015.