**Homework 5 Report**

Bin Gao & Bin Yan

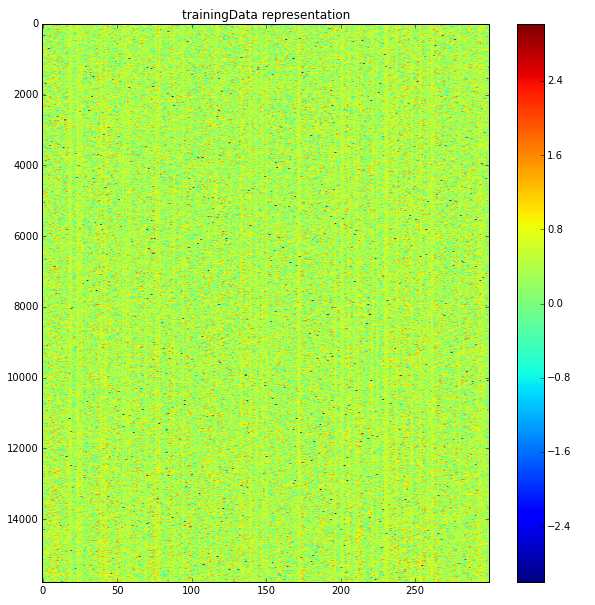
**Problem**

Multi-class classification of test data based on training data and training labels. There are 300 features in total and around 15000 training samples with some missing data.

**Our Approach**

*Dealing with missing data*

The trainingData.txt consists of a matrix of size 15770x300. Among all the data points in trainingData.txt, 9495 of them are missing data points. Here is an image representation of trainingData.txt:

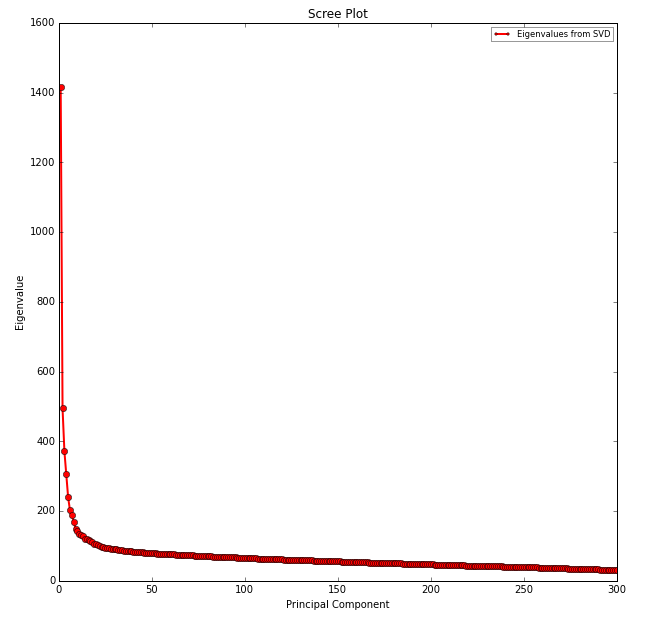


Very dark dots in the image are missing data. We can see from the image that the missing data are sparse.

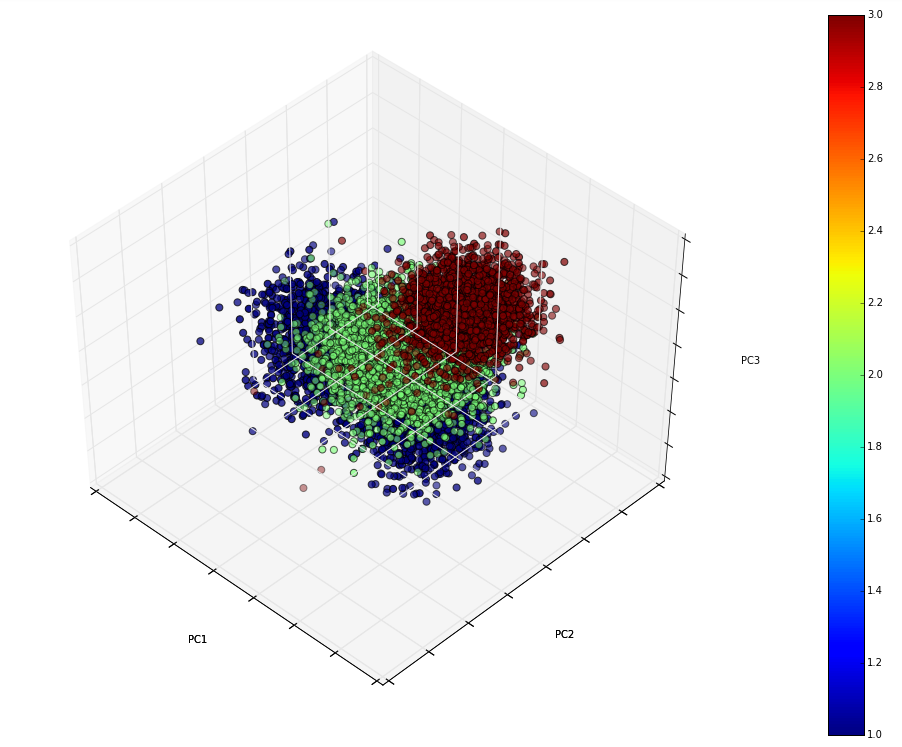
We first tried using the median of the whole column (feature) to represent the missing data. Later we have tried discarding all rows (data instance) containing any missing data to train the model, which will leave us with only 8673 of 15770 data instances. These 2 methods will either add noises or waste data instances. In the end, we decided to group all data instances into 3 groups based on training labels. Within each group, we use the median of columns to represent the missing data. Then merge 3 groups back into one training data. We also discard all data instances that contains more than 1 missing data point.

*Feature reduction*

There are 300 features given by the data. The Scree Plot shows us that not very feature is important on determining data’s classifications:



Based on the scree plot, there are probably less than 25 principal components that have very high Eigenvalues. Thus, we picked 30 components as the default output components from PCA transformation. In fact, if we try to visualize the data by using first 3 principal components, they are already very separatable visually:



We tested the classifiers with and without PCA. Random forest results are very sensitive to PCA. Without PCA, the prediction accuracy of random forest is around 0.65. With PCA, the prediciton of random forest can increase to 0.95. So PCA can greatly improve Random Forest accuracy. PCA does not have that large impact on other classifiers.

We have also tried auto-encoder from scikit-neuralnetwork package to do the dimension reduction. We didn’t get good results from using auto-encoder, so we decided not to use auto-encoder for dimension reduction. PCA, however, has consistently performed well on different classification algorithms.

Since from analysis, we know that not every feature is important, but we certainly don’t want to discard any feature. Therefore, we use PCA to reduce 300 features into 30 principal components. There 30 principal components will have more or less weights of each feature in the original 300 features. They will help us better predict the correct labels. But because of using PCA, we have no idea on what specific subset or any individual feature of the 300 features is important.

*Algorithms and in-code representation*

Logistic regression

clf1 = LogisticRegression(random\_state=1)

Random forest

clf2 = RandomForestClassifier(random\_state=1, n\_estimators=20)

Gaussian Naive Bayes

clf3 = GaussianNB()

Decision tree

clf4 = DecisionTreeClassifier(max\_depth=4)

K-neighbours

clf5 = KNeighborsClassifier(n\_neighbors=7)

Support vector classifier

clf6 = SVC(kernel='rbf', probability=True)

Adaboost

clf7 = AdaBoostClassifier(random\_state=1)

Neural network

clf8\_1 = Classifier(

layers=[

Layer("Maxout", units=100, pieces=2),

Layer("Softmax")],

learning\_rate=0.001,

n\_iter=25)

We used default values for neural network parameters, and it’s performing very well.

Since sklearn doesn’t come with a good implementation of neural network, we picked scikit-neuralnetwork as our package for Neural network and is also compatible with other scikit-learn package and its VotingClassifier. After adding third-party neural network package into our program, we have covered all major classification algorithms in the course.

We tried soft voting with and without weights. We also tested Multi-class classifiers and binary classifiers. For binary classifiers, we output the probabilities of a data point being in this class versus in the other classes. The class with the highest probability is assigned as the predicted class. Our final submission is based on ***binary*** classification. The final class and the first 3 columns for each data instance in the test data is produced by using the binary classifiers.

*Version requirements for packages*

Almost all the classification algorithms are from sklearn. You need to upgrade sklearn to version 0.17-np110py34\_1 in order to run soft voting.

The neural network classification package is from scikit-neuralnetwork, [http://scikit-neuralnetwork.readthedocs.org/en/latest/index.html#](http://scikit-neuralnetwork.readthedocs.org/en/latest/index.html), which could be easily installed by running “pip install scikit-neuralnetwork” from a OSX terminal.

*Cross validation*

We performed 10-fold cross validation for individual multi-class classifiers in order to get an idea of their performance. Based on the cross validation results, we can assign a higher weight for the classifiers with excellent performance. Table 1 shows the 10 fold cross validation results of ROC AUC scores, as well as prediction accuracy based on final label prediction. PCA is not performed for this cross validation.

Table 1 ROC AUC scores comparing all approaches based on training data with no PCA

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Classifier | ROC AUC score | | | | Prediction accuracy | |
| 1 | 2 | 3 | Average | Mean | Std |
| Logistic Regression | 0.96 | 0.92 | 0.96 | 0.95 | 0.84 | 0.008 |
| Random Forest | 0.87 | 0.74 | 0.84 | 0.82 | 0.65 | 0.011 |
| Gaussian Naïve Bayes | 0.97 | 0.94 | 0.97 | 0.96 | 0.86 | 0.007 |
| Decision Tree | 0.70 | 0.60 | 0.70 | 0.67 | 0.49 | 0.015 |
| K Neighbour | 0.96 | 0.84 | 0.93 | 0.91 | 0.72 | 0.010 |
| Support Vector Classifier | 0.97 | 0.94 | 0.97 | 0.96 | 0.86 | 0.010 |
| Neural Network | 0.99 | 0.98 | 0.99 | 0.99 | 0.95 | 0.060 |

The results show that logistic regression, Gaussian Naive Bayes, support vector classifier and Neural Network have very good performance in terms of ROC AUC score. Neural Network has the highest prediction accuracy and AUC score. Therefore, we will have high weights for these classifiers. The performance of decision tree is not satisfying, and we removed it from our ensemble classifier.

**Testing Results**

All testing results after evaluation software bugs are fixed are below, most of the testing results before the bugs are discarded because of wrong indications.

Table 2 Testing results

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Description | Weights | | | | | | | ROC AUC score | | | |
| lr | rf | gnb | kn | svc | ab | nn | 1 | 2 | 3 | Avg |
| binary, soft voting, NA=111 filled by median of all data | 1 | 1 | 1 | 1 | 1 | 1 |  | 0.983 | 0.954 | 0.973 | 0.970 |
| multi-class, svc, NA=002 filled by median of that label |  |  |  |  | X |  |  | 0.986 | 0.962 | 0.976 | 0.975 |
| binary, svc, NA=003 filled by label median |  |  |  |  | X |  |  | 0.986 | 0.958 | 0.974 | 0.973 |
| multi-class, rf, NA=111 filled by label median |  | X |  |  |  |  |  | 0.895 | 0.751 | 0.855 | 0.834 |
| multi-class, nb, no PCA, no NA filtering, filled by label median |  |  | X |  |  |  |  | 0.974 | 0.944 | 0.967 | 0.962 |
| multi-class, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  |  | 0.985 | 0.960 | 0.974 | 0.973 |
| binary, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 | 1 |  | 0.984 | 0.956 | 0.973 | 0.971 |
| binary, soft voting, NA=002 filled by label median | 1 | 1 | 1 | 1 | 1 | 1 |  | 0.982 | 0.953 | 0.973 | 0.969 |
| multi-class, soft voting, NA=002 filled by label median | 1 | 1 | 1 | 1 | 1 | 1 |  | 0.983 | 0.957 | 0.974 | 0.972 |
| binary, soft voting, NA=111 filled by label median | 1 | 2 | 1 | 1 | 2 | 1 |  | 0.984 | 0.957 | 0.973 | 0.971 |
| binary, soft voting, NA=111 filled by label median | 1 | 1 | 1 | 1 | 1 | 1 |  | 0.984 | 0.955 | 0.973 | 0.971 |
| multi-class, soft voting, NA=000 | 1 | 1 | 1 | 1 | 1 | 1 |  | 0.984 | 0.957 | 0.973 | 0.971 |
| multi-class, nn, NA=000 |  |  |  |  |  |  | X | 0.982 | 0.948 | 0.965 | 0.965 |
| binary, nn, NA=002 filled by label median |  |  |  |  |  |  | X | 0.984 | 0.950 | 0.971 | 0.969 |
| multi-class, NA=000, auto-encoder |  |  |  |  |  |  | X | 0.866 | 0.587 | 0.877 | 0.776 |
| multi-class, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  | 2 | 0.986 | 0.961 | 0.975 | 0.974 |
| multi-class, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  | 3 | 0.986 | 0.962 | 0.975 | 0.974 |
| binary, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  |  | 0.984 | 0.956 | 0.973 | 0.971 |
| binary, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 4 |  | 3 | 0.986 | 0.959 | 0.975 | 0.973 |
| binary, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  | 4 | 0.986 | 0.958 | 0.975 | 0.973 |
| multi-class, soft voting, NA=111 filled by label median | 2 | 3 | 2 | 2 | 3 |  | 4 | 0.987 | 0.962 | 0.975 | 0.975 |

lr: logistic regression

rf: random forest

gnb: Gaussian Naïve Bayes

kn: K-Neighbours

svc: support vector classifier

ab: Adaboost

nn: Neural Network

All of our submitted classifiers (except random forest without PCA) have quite good performance. Multi-class classifiers have consistently slightly better performance over binary classifiers. But we submitted the final results and corresponding code based on binary classifiers as required.

**Conclusion**

The final version of our binary classifier using soft voting of the following classifiers and weights.

* Logistic regression

Weight given to VotingClassifier is **2**.

* Random forest

Weight given to VotingClassifier is **3**.

* Gaussian Naive Bayes

Weight given to VotingClassifier is **2**.

* Decision tree

The AUC score is low, so we didn’t put it into our VotingClassifier.

* K-neighbours

Weight given to VotingClassifier is **2**.

* Support vector classifier

Weight given to VotingClassifier is **3**.

* Adaboost

As the adaboost is very slow and the testing results are not satisfying, we drop this classifier. Weight given to VotingClassifier is 0.

* Neural network

Weight given to VotingClassifier is **4**.

**Instruction of running the code**

There are needs to upgrade scikit-learn to 0.17-np110py34\_1 and install scikit-neuralnetwork before executing the program. Please look at section “Version requirements for packages” for more details.

The program can be executed by supply one command line argument, the test data filename, to the main executable. The training data and labels are always considered to be “trainingData.txt” and “trainingTruth.txt” in the current directory, which means they are hard-coded in the program currently. During the execution, there will be UserWarning generated depending on the versions of the installed package in your system. The program will finish its execution regardless whether some UserWarnings are generated. The resultant test labels will be stored in y1\_binary\_eclf\_with\_weights2322304\_NA111.txt of the current directory.

Sample command to obtain output:

$ python classifier\_binary.py testData.txt

Sample result file generated in the current directory:

y1\_binary\_eclf\_with\_weights2322304\_NA111.txt

**Appendix**

“Back to basics.ipynb” contains some experimental code on how to generate the image used in this write-up and how to generate some metrics on choosing weights on each classifier we used.