IN4320 - Machine Learning

Final Assignment

Wei Sun 4711114

October 19, 2018

1 Introduction

The provided dataset consists of 200 labeled samples with 100 samples per class and 20000 unlabeled samples, and each sample has 204 dimensions with certain noise. Compared with dimensions, labeled dataset is really small and it is less feasible to get good performance using complex classifiers or those which are likely to suffer Curse of dimensionality like KNN classifier. In this situation, I am going to mainly investigate linear classifiers like Linear Discriminant Analysis and Linear SVM. Besides, to make use of those unlabeled data, I would use semi-supervised learning techniques to improve the performance of base classifier trained by labeled data.

The experiments are conducted by using Python with some popular libraries for Machine Learning such as scikit-learn and numpy.

2 Data Analysis and Pre-processing

2.1 Feature analysis

The samples have relatively high dimension(204),so before training classifiers I use PCA to analyze these features to see if we can reduce the feature dimension. Figure 1 shows the sum of PCA explained variance ration of different components, we can see that 150 components or features can explain around 95% , so we can do a feature reduction via PCA from 204 features to 150 features per samples.

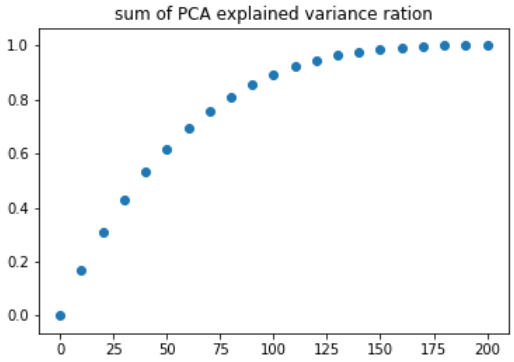

Figure 1: sum of PCA explained variance ration v.s. number of components

2.2 Create train,validation and test dataset

The 200 labeled samples are split into three datasets which are train set(50 samples per class), validation set(25 samples per class) and test set(25 samples per class).

Train set is used to train base classifiers and the test accuracy on validation set will be used during tuning phase of semi-supervised learning. Test set plays as unseen data and evaluate the 'true' performance of tuned classifiers.

3 Evaluate candidate classifiers

As mentioned above, I will mainly focus on two classifiers, LDA and Linear SVM. In this section, these two candidate will be evaluated based on labeled dataset and the better one will be selected as base classifier for further semi-supervised learning.

The evaluating phase is quite simple, after training by train set, validation set is used to test the performance of these two classifiers.Figure 2 and 3 shows the performance of LDA and Linear SVM on this dataset respectively. Comparing these two figures we can find that Linear SVM is better than LDA in this dataset because it achieves higher accuracy.

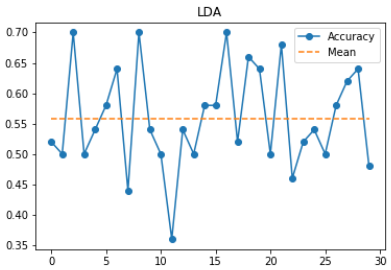

Figure 2: performance of LDA

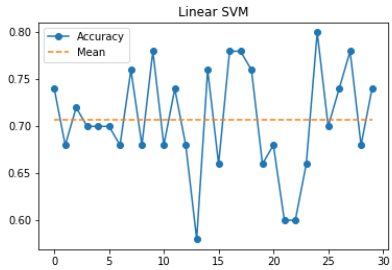

Figure 3: performance of Linear SVM

4 The variation semi-supervised learning

Before going into semi-supervised learning, we should take a look at our unlabeled data. Figure 4 shows the histogram of the distance of unlabeled samples to the decision boundary of base classifier. The x axis means the distance and y axis shows the number of samples belonging to corresponding distance groups.

We can see that the distance of most samples is less than 0.5, and these close samples are which we should make use of to fine-tune our base classifier. That makes sense because samples far away from decision boundary would have less influence of our classifier.

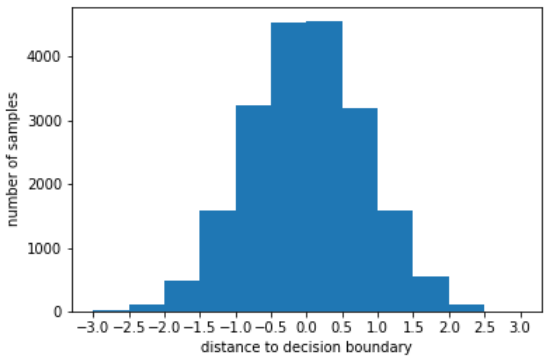

Figure 4: Histogram

So, how to make use of these unlabeled data to improve our base classifier? The main idea is to use the classify accuracy on validation set to determine if the new coming labeled samples which are labeled by base classifier will be kept or not. Section 4.1 shows the details of the algorithm

4.1 Algorithm of the semi-supervised learning

1 Train base classifier using train dataset

2 Use the base classifier to classify the unlabeled data as well as compute the their distance to decision boundary.

3 choose **n** new labeled samples per class which are close to the distance boundary(maximum distance is user-defined **t**), where **n** is a user-defined hyper-parameter, and add them to the train set to re-train the classifier.

4 Test the accuracy rate on **validation set**, if the accuracy rate increases or the accuracy rate is higher than **p** which is also a user-defined hyper-parameter. The new adding samples will be kept, otherwise the samples will be removed and use old data set the re-train the classifier.

5 Repeat step 2-4 **k** rounds where **k** is a user-defined hyper-parameter.

Figure 5 shows the plots where x axis means the number of round and y axis means the error rate. The estimated accuracy rate is around 0.75 and error rate is around 0.25.

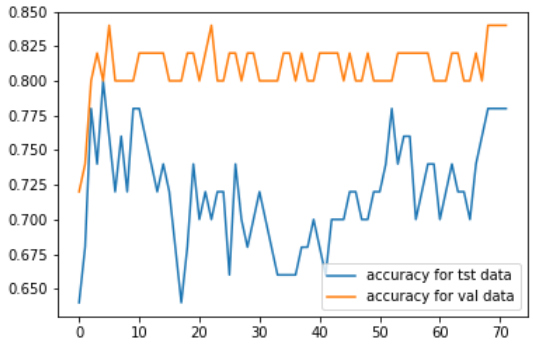

Figure 5: accuracy for

5 Discussion

This semi-supervised learning algorithm is inspired by self-train. I made some modifications for the linear base classifier. However this variation brings more hyper-parameters which should be chosen carefully. The distance threshold **t** is the most important parameter, if it is too large, then the difference of accuracy rate between validation set and test set would increase which is caused by over-fitting. When choosing a small threshold **t**, it will be less likely to cause significant over-fitting but the optimization time or the time of finding satisfied samples will increase.

During my experiments, I found that this algorithm does not guarantee convergence. After adding more samples to train set, the accuracy rate on validation set would be high but on test set the accuracy rate will decrease significantly. I think this is also caused by over-fitting, so choosing a reasonable stopping round is very important.

Overall, this algorithm does improve the performance on test set, and it is able to benefit from new samples. However, due to the limitation of time, I can not do more experiments or try different base classifiers to find a mathematical explanation behind this method that how these hyper parameters influence the performance and why it works(or maybe not work for true accuracy tested by the company).

Around 850 words in this report