**EE 236 Project: Multiclass Classification**

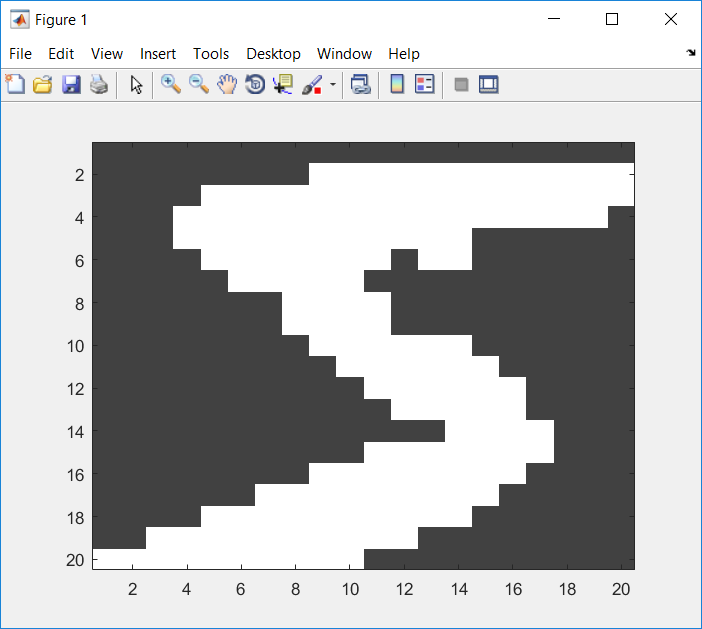
**Fangyao Liu 204945018**

1. Description

The goal of this project is to utilize a linear classification approach to create a multi-class classifier. In this report, we take M-NIST data set, which is a collection of 20 X 20 images of handwritten digits from 0 to 9, as training/testing data and utilize Support Vector Machine (SVM) as the tool to decide separating hyperplanes. According to the requirement, we are asked to present two classifiers. One of them is the best classifier we can come up with, another one should take fewer hyperplanes. In order to accomplish this requirement, this report presents two method that are often used in multiclass classification called “one vs one” (ovo) and “one vs rest” (ovr). Also this report presents how a “dimension-reduced” data will influence the result.

1. Data Observation

Before taking any further steps, first observe the data set. Project package contains two data sets. One is MNIST\_Train.mat and another is MNIST\_Test.mat. According to project introduction, the data is a 20 X 20 image of handwritten digits from 0 to 9.



*Image of handwritten “5”*

The training features are a matrix of 400 X 60000, each column represents an image. The training label is a vector of 60000 X 1, containing the corresponding digit. Testing features and labels are the same but of 10000 data points. Considering the size of the training data and laptop’s computing power, this report presents the experiments of only taking 6000 original images out of 60000 as training points.

However, a dimension reduced version of data can be computed by averaging a 2X2 cube into 1 element. In this way, 20X20 image turns into a 10X10 image and we still keep the shape of hand-written digit. The results of original data and dimension reduced version of data will be presented in the testing result as a comparison.

1. Training Classifier

Support Vector Machine is known as a famous supervised learning model used for classification problem. Because solving the hyperplane of support vector machine is essentially solving a linear programming problem, here we set up a second norm and soft margin model and then use CVX as solving tool to find the separating hyperplane.

**Equivalent LP:**

* One vs One

In “One vs One” algorithm, we train hyperplane group by group. For example, we first pick all the data points in group “digit 3” and group “digit 7”. Then using SVM to find a hyperplane to separate these two groups and keep on doing this for the test possible group combinations. In this way, we are able to find a multiclass classification system combined by several hyperplanes. In our project, the total number of groups are 10, so the total number of combinations of hyperplanes is 45. After deciding the hyperplanes, we can test the points by throwing it into all these hyperplanes. Each hyperplane will give a vote to certain group and final prediction will be the group with the most number of hyperplane votes. According to the project requirement, this is the most accurate algorithm that this report is going to present, so it is implemented in “MyClassifiier1.m”.

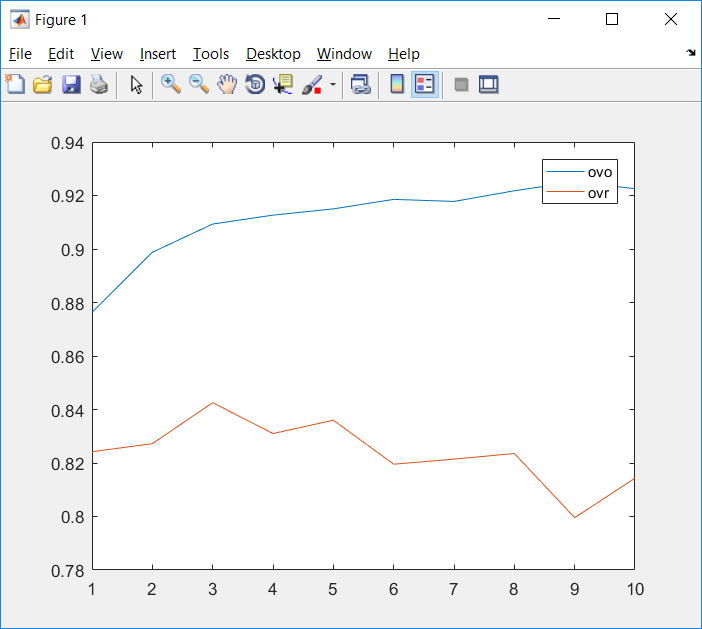
* One vs Rest

“One vs Rest” algorithm presents a way that has fewer hyperplane than the first one. The idea of this algorithm is first setting one of the group as “1” and the rest groups as “0”, then solving a linear programming problem and find a hyperplane. In this algorithm, the number of the hyperplane will be the same as the group numbers, so there will be totally 10 hyperplanes in this project. Compared to 45, this is a great decrease. However, as a compensation, the accuracy of the prediction will decrease at the same time. According to the project requirement, this is the classifier with fewer hyperplanes, and it is implemented in “MyClassifer2.m”.

1. Testing result

After implementing and training the classifier, a testing result should be presented as a direct way to measure the performance of the classifiers. Here, the project will present a 10-fold cross validation to compare the performance of two classifiers, then compare the performance of two classifiers on test data set.

For original data:



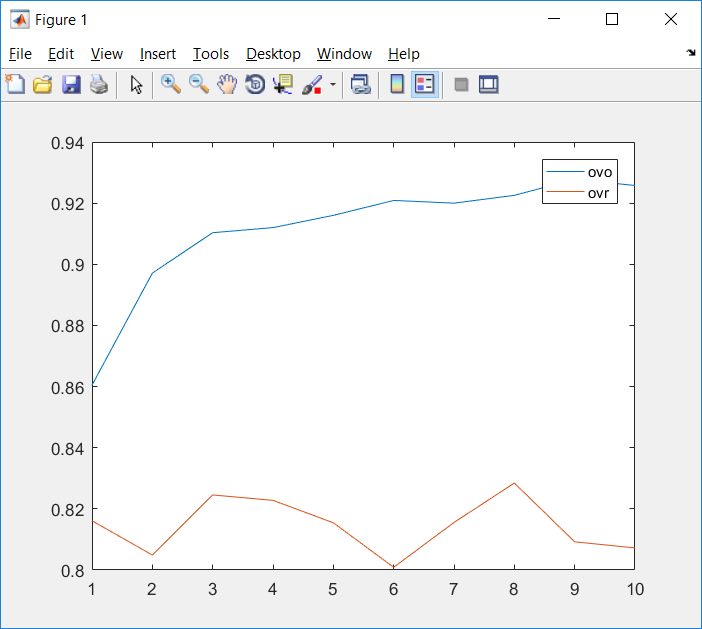
*Each fold’s precision result of original data*

As expected, “ovo” has a better result than “ovr” because of the “hyperplane trade off”.

Then use 6000 training data to train two classifiers and test on all the data set. The result is shown as below:

|  |  |
| --- | --- |
| One vs One | 0.8779 |
| One vs Rest | 0.7813 |

For dimension reduced data:



*Each fold’s precision result of dimension reduced data*

As we can see over here, the dimension reduced version didn’t bring a dramatic decrease on precision. It implies that we the way we use to reduce dimension still keeps the essential information of data. Besides, the time that the codes take has been shortened a lot!

Then use 6000 training data to train two classifiers and test on all the data set. The result is shown as below:

Dimension reduction result

|  |  |
| --- | --- |
| One vs One | 0.9045 |
| One vs Rest | 0.9026 |

1. Conclusion

After getting the error rate on cross-validation and testing set, we can reach a conclusion that “one vs one” has a better classification ability than “one vs rest”. However, this is achieved by computing more hyperplanes. There is trade-off between the number of hyperplanes and accuracy. Besides, a dimension reduced data saved a lot of computation when we are doing multiclassification.