Johnny Chen

Machine Learning

Professor Keene

KAGGLE REPORT #1

For the Kaggle Competition, we were asked to classify data knowing that there are 561 features and 12 classes. I used Python for this Kaggle because I know that if you import the right packages, it would be easy.

So the first step for any of the approaches was to import the data from the test files. Then I created a classifier that would fit the training features to the training labels. Then it would simply try to classify the test data and return some results

The first approach I tried was to get a classifier that was using Gaussian Naïve Bayes. Before I submitted my file, I ran it and I got weird numbers in the CSV file – numbers that did not match what I saw in the example file. It turned out that I was writing the test features to the file and therefore I was getting multiple numbers on each line, each which was a long decimal. When I finally realized that I should just have logged the “i" value, I thought that the results would be decent. However, I got 0% when I submitted my results so I decided that the Gaussian approach was not a good one. I later learned that this was probably due to how I formatted my file.

Afterwards, I decided to use SVMs, short for Support Vector Machines because supposedly they are better at classifying data. Also we have a lot of data so even though we have 561 features it should work fine.

I got 0% each time I submitted my results and it seemed really weird why this was so. I asked some classmates for their results and compared with what they had. It seemed that my results were for the most part extremely close to what they got. But why did I do so bad? After looking at my code for a while, the most likely explanation turns out to be that when I was writing to the CSV file, I put a space after each comma, and when I compare this to the example file, there is

For next time, I would like to look into different kernels. By default for the SVM kernel, “rbg” is used. Maybe other types like “poly” would work better. Also, we can play around with the kernels and specify them to our needs, so SVMs are extremely versatile. Also I would like to take the test data and try to split it. I think this is what cross validation is. That way I would have less training data but given the vast amount of data, it would still be a lot of training data. So even though I have less data to train with, I am able to have more data to test with. If I can make some accuracy function, I can then get a sense of how accurate the classifier is without having to go onto Kaggle to check, which only lets you submit twice a day.