EECS 349: Eye tracking biometrics **Group 37: Lauren Hutson and Michael Tobia**

Problem definition:

The task we chose is to create a learner that can accurately predict who a person is based off of the movements of their left and right eyes when tracking an object on a screen. Our goal is to create a learner that has a better accuracy than the winning learner of the EMVIC 2012 competition, which was 39%.

Dataset:

The dataset we're using was provided by the EMVIC 2012 competition whose link can be found at the bottom of this page. The dataset consists of 978 samples taken from 37 different people. Each sample contains 8192 attributes which come from the left and right eyes' x and y positions when gazing at a moving object on a computer screen during each testing session. The dataset is stored as a csv that we load into our python script. The script then splits each row in the following order: class, left x-dir for all tests, left y-dir for all tests, right x-dir for all tests, and right y-dir for each timestep during the test. Figure 1 below shows an example plot of one participant's recorded eye movements.

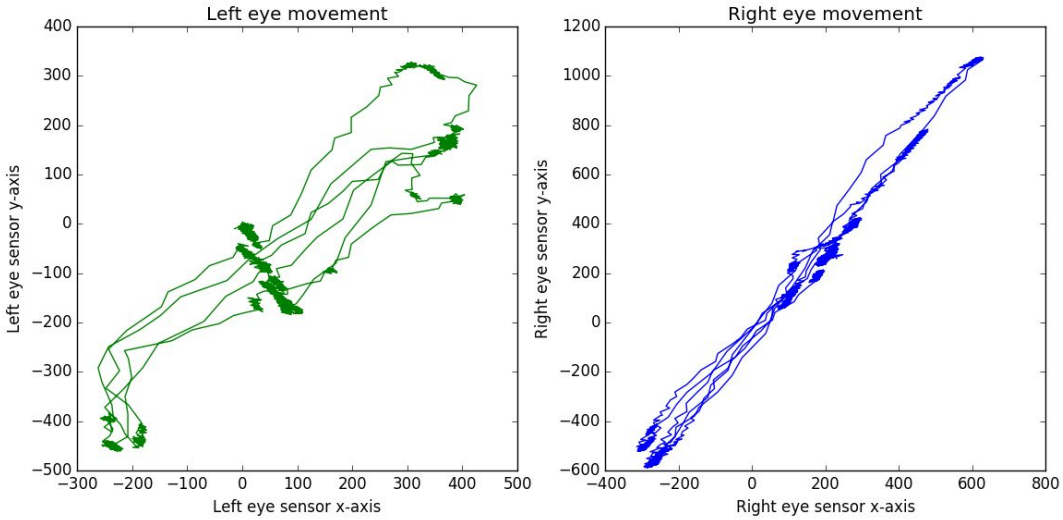

Figure 1: This plot shows the recorded eye movements of one participant from the EMVIC dataset. This participant was labeled #31.

Testing:

To train our learner we are using the EMVIC 2012 training dataset found on kaggle along with 10 fold cross validation. Because the datasets are split into multiple trials of the 37 participants, using 10 fold cross validation we will be able to train on data from all 37 of the participants, with about 10% of the data randomly selected for testing on each fold.

Results:

To begin, we obtained a benchmark accuracy using wekas ZeroR classification to find the lowest acceptable classification percent in order to judge the performance of our other learners. Using ZeroR we were able to get correct classification of 16.1043% of our testing data.

We first attempted to classify the data using a neural network. This was our original method of training on the dataset and, as of the midpoint of the project, we were obtaining 15% accuracy on the testing set. This is worse than the ZeroR results, which signifies that our classifier didn't actually do any classification. To try to increase our accuracy to be closer to that of the winner from the EMVIC competition we utilized the following methods: increasing the number of hidden layers in the neural network, varying the activation functions and number of nodes used in each layer of the neural net, and adding dropout to each layer.

When increasing the number of hidden layers from 1 to 3 with 8,5, and 3 nodes in each respective layer, the network's accuracy went up by 5% giving a final training accuracy of 20%. This new accuracy is still well below that of the winning accuracy of the competition but much better than the original accuracy of the neural net. With each layer that is added to the network the dimensionality of the network increases thereby enabling it to create a greater number of relationships between the data and the classes compared to the smaller network. Although, considering the number of nodes in each layer and the size of the dataset, there is a worry that the network does not have enough memory to learn weights of all of the attributes, and only learns a small fraction. However, increasing the number of nodes in each layer worsened the performance of the classifier, so we decided to continue using the smaller network.

When changing the activation functions of the input and hidden layers of the network to softmax or Sigmoid the difference in accuracy was imperceivable. There was however a very slight increase of less than a percent when we changed the output layers activation to softmax.

With the addition of a dropout layer with value of .3 after the first hidden layer the accuracy increased from 20% to around 24% and another layer with value of 0.1 after the second hidden layer allowed for an increase of almost 2% leaving use with a final classification accuracy of 25.92%.

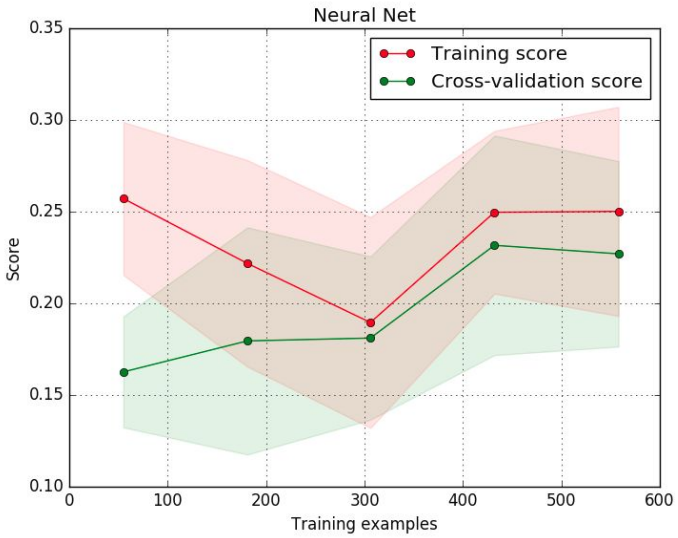

Figure 2: This is the training and cross validation accuracies of the Neural Network. The faded red/ green shaded areas show the maximum positive and negative variance of the data. The solid line is the median value.

As an experiment we also decided to pass the data through two tree based learners, decision trees and random forests. The decision tree generated by weka was used to see how the data gets classified within a tree structure. Using a decision tree without pruning we were able to get 36.503% of the validation set correctly classified. This result is within a very close range of the winning percentage for the EMVIC competition that we originally pulled the data from. This percentage increased very marginally to 36.65% when using the pruned model, which cut the decision tree size down from a size of 245 to 241. The random forest, which was generated with a seed value of 1 and a batch size of 100, by far gave the best classification accuracy at 60.88%. This classifier required very little modifications in order to get a vastly improved result. We think that the tree based learners can get vastly improved accuracy because, during eye movement, if a person sees the same visual data during each test, their eye movement will be the same across the different testing sessions. Figure 3, below, shows a visualization of the decision tree generated by weka.

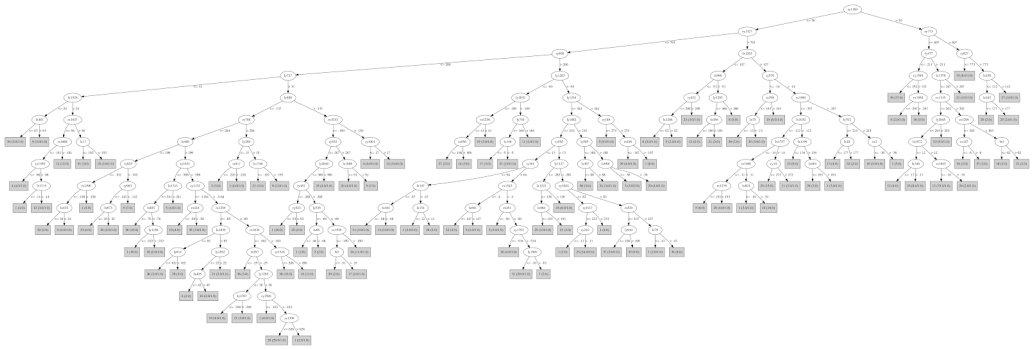

Figure 3: This is the decision tree generated by weka. This shows what attributes are being considered to contain the most information about each person. A high res version of the image can be seen here.

In our final experiments we looked into using scikit-learn's random forest and support vector machine capabilities. Scikit-learn's library provides many parameters for both classification methods, and, since they were used as benchmarks on the original EMVIC competition, they were a good match for our dataset. Curiously, scikit-learn's random forest performed better than weka's, achieving an accuracy of 64.72% using 10-fold cross-validation and default settings. By default, scikit's random forest only considers a number of features equal to the square root of the total number of features for each split. In our case, this number is 90. By increasing this number to 15% of the total features (1228 features considered per split), we were able to increase the accuracy of this model to a final 65.537%. Increasing considered features beyond 15% both reduced accuracy and massively increased processing expense.

Scikit-learn's support vector machine (SVM) supports different kernel functions. By default, SVM achieved 64.39% accuracy. Changing the kernel function to a second degree polynomial allowed the model to achieve 68.72% accuracy. Finally, by changing the SVM kernel function to a sigmoid profile, the classifier was able to achieve 65.602% accuracy.

From the results gathered we can come to the conclusion that the best learners out of those that were used to classify the dataset are the random forest and support vector machine. As stated before, we were able to get 65.537% and 68.72% accuracy from the learners respectively, which is significantly higher than that of the neural network but, it is much higher than the winning accuracy of the competition. If we were to continue working on solving this problem I believe that we would continue to pursue SVM or tree based learners such as the Decision tree or random forest because they give vastly improved accuracy when compared to the neural network. Along with this, datasets for EMVIC 2014 competition do exist, though they are difficult to obtain. Getting this dataset

would provide a much larger dataset to work with, and likely boost accuracy for all learners.

Work Distribution:

For the project the distribution of work is as follows. Lauren worked on creating the Decision Tree, and putting together the Neural network, and the website. Mike assisted in experimenting with various learners and editing documents.

Links
https://www.kaggle.com/c/emvic/data