James Alford-Golojuch

CS460G

December 3, 2016

Assignment 5 Report

Round 1:

For round 1 I used python's sklearn MLPClassifier with the same tuned parameters that I successfully used in the last assignment with the only difference being that for solver I used 'adam' instead of 'sgd' since 'sgd' gave me a score of around 4.4 with my first submission to Kaggle with this model. This model is a neural network called a Multilayered Perceptron that optimizes the log loss function using the solver and for that solver I used adam which is a stochastic gradient based optimizer. For the activation function I used 'relu' which is a rectified linear unit function. I used the default value for the number of hidden layers which is one hidden layer with 100 nodes as that seemed to do pretty well in my testing last assignment. I also used the given features and for the output I got the probabilities for each class rather than setting the most likely class to a value of 1 and 0 for the rest. This use of actual probabilities rather than just picking the best option and setting the probability to 1 is consistent throughout all of my rounds. I ended up with a score of 0.16774 which as it turns out would be the best performing model for me. Also in comparison to other posts on the discussion boards for all of the rounds at the time I am writing this it appears to be one of the better performing models. From this round I learned that something as simple as one parameter change can drastically change the effectiveness of the MLP classifier which is what I experienced with the change in the solver from a pure sgd solver to one that is based on the same solver but optimizes differently.

Round 2:

For round 2 I used python's sklearn RandomForestClassifier with a combination of the default parameters along with changes to the parameters of n_estimators and max_depth which I based off of the submission from Anton who used a similar model in the Round 1 discussion. I kept the n_estimators = 150 as per his test. I did double the max_depth value to 50 which I found to help slightly improve the performance of my test versus when I used his tested value of 25. I attempted to change a few other parameters from the default settings but I found that they either did not help my score or actually made it worse. As I did with the first round I used the given features of the images rather than working with the images themselves. I found the Random Forest is easier to setup and the model trains and tests a lot quicker than the neural network model I used in the 1st round. This speed up did come at a cost though as I was unable to approach the score I received in round 1. I ended up receiving a score of 0.68502 for this submission which is significantly worse than my neural network from Round 1 but ultimately proved to be my second best model used. As noted above I learned that the Random Forest is a

lot faster and easier to setup and test than the neural network which comes with a cost of a worse score. When choosing a model you have to balance the performance you want vs the time you have to train and test the model

Round 3:

For round 3 I used python's sklearn KNeighborsClassifier and started with the default parameters of uniform weights, a leaf size of 30 and the distance metric minkowski using Euclidian distance the score was similar to my final score but I was able to slightly improve it by switching the distance weights from uniform to having those neighbors closer to the point having a greater weight. I also attempted to increase the number of leafs but that did nothing. I was able to have a better score using 9 nearest neighbors rather than only using the default value of 5. Compared to my other models I used KNN was able to be calculated the fastest but it also had the worst results which is consistent with my comparison between rounds 1 and 2. The models that are the fastest to compute tend to be the worst scoring ones as seen with the other two rounds. I ended up with a score of 1.03376 which though is higher than my other tests did surprise me that it didn't perform worse since from looking at other people's posts on the previous rounds it appeared that this method would be a lot worse than my other previous ones. What I learned was consistent with what I learned from the other rounds which was that the faster a model is able to be computed the worse the results tended to be.