Dustin Casey

CS 167 Write-up

11/21/2016

**Project 4: Face Recognition Write-Up**

**First Attempts at Facial Learning with ANN:**

I started out using the Artificial Neural Networks algorithm which utilizes Multi-Layer Perceptron’s to see how well this algorithm would work at recognizing and learning faces. I wanted to train the algorithm to learn the features of the face, remember each of the faces and the components of them, and be able to guess correctly a new face. Using only people with at least 70 faces in our data set, I split the data into train\_data, test\_data, train\_target, and test\_target so that we would have predictors and target columns for use in our algorithm. I used the MLPClassifier algorithm and fitted the train\_data and train\_target, predicted for test\_data and compared the predictions to the actual test\_targets to get an accuracy score. The ANN was not very successful at learning faces and was very inaccurate with an average accuracy score of approximately 48%.

**Using Support Vector Machines and SVC:**

After using ANN and seeing that is was not as effective as it could be, I decided to try using a Support Vector Machine, specifically the classifier algorithm (SVC). Using the SVC, the score improved to the 80s range without changing any of the parameters. I then wanted to get an idea of which kernels would be the most accurate and efficient at learning faces. I created a new SVC for each kernel and printed out the accuracy results. After running the code multiple times, the top two kernels were ‘linear’ and ‘poly’, with average scores at approximately 85%. ‘rbf’ and ‘sigmoid’ did not do so well with learning on this original and typically had an average score of approximately 43%.

**Using Principal Component Analysis:**

After seeing the improved results with Support Vector Machines compared to the results I received with ANN, I decided to use Principal Component Analysis (PCA) in an attempt to reduce the number of attributes used from 2914 to around 100. Using PCA, I adjusted the n\_elements to 100 so we can get our 100 attributes we are looking for and set whiten to True to help increase the accuracy of the algorithm. After fitting the train\_data and transforming the data for both the train\_data and test\_data, I used SVC again for each of the kernels to see if there was any improvement in accuracy scores after doing PCA. I found that both ‘sigmoid’ and ‘rbf’ kernels improved in accuracy immensely and had a new average accuracy of approximately 82% for ‘sigmoid’ and 83% for ‘rbf’. At the same time, both ‘linear’ and ‘poly’ kernels decreased in accuracy, although ‘linear’ only decreased slightly to approximately 75% while ‘poly’ decreased quite a bit to approximately 45% average accuracy.

**Using GridSearch:**

After implementing PCA to reduce the number of attributes we are using for learning, it was time to implement GridSearch, a tool used to compare different parameter values to one another with a parameter set and across sets. I wanted to see which parameter values work best with each other and with each different type of kernel and then compare all of those accuracies to find the best combination that will result in the most accurate algorithm. GridSearch makes with process easy by doing it for me. I used 9 different C values (.00001, .0001, .001, .01, .05, 1, 10, 50, and 100), 9 different gamma values (100, 10, 1, .01, .001, .0001, .00001, .000001, and .0000001), 5 different degree values (used only for ‘poly’; 1, 2, 3, 4, and 5), and 4 different kernel values (‘linear’, ‘poly’, ‘rbf’, and ‘sigmoid’). Although I have changed the order of the data multiple times and rerun the code repeatedly, some similarities I have found include: ‘rbf’ is consistently the most accurate kernel after using PCA with average accuracy scores of approximately 87%; when the gamma is medium-to-lower, the accuracy score tends to be higher (usually got a gramma of .01 or .001); when C is larger, the accuracy score tends to be higher (usually had a C of 10 or 50); degree seems to consistently be best at 3, which is the automatic preset parameter value.

**Conclusion and General Insights:**

Overall, using SVC, PCA, and GridSearch helped me to find the most accurate parameters and algorithm to learn and predict faces. Without using PCA and just using SVC, the kernels for ‘linear’ and ‘poly’ were more accurate than the others. But, when implementing PCA and using SVC, the kernels for ‘rbf’ and ‘sigmoid’ tended to be more accurate than the others. Grid search helped me to find the best parameters for learning and predicting, leading to the best accuracy out of all of the tests I ran. Although the best parameters changed because of the multiple times I reran the code and the randomness of the shuffling, some general themes I saw was that C value tends to be more accurate when it is a bit higher number (1, 10, 50) and gamma tends to be more accurate when it is lower or middle of the road (.01 or .001). ‘rbf’ tended to be the most accurate kernel after PCA and it was the highest overall accuracy after all of the tests. With PCA, SVC ran much, much faster than without.