Brianna Shade

CS545: Machine Learning

2/1/14

Project 2: Dogs vs. Cats with SVMs

OVERVIEW

This project took a look at distinguishing pictures of dogs from those of cats through the use of support vector machines. Each color photograph was reduced to 64 features, each representing a color value of a group of pixels across the picture. These features were then used to train a support vector machine model using SVM-Light. Finally, a separate test set was processed to achieve the final accuracy scores.

LINEAR AND POLYNOMIAL CROSS-VALIDATION

To implement cross-validation, the training data was first divided into 10 equal parts, each consisting of an equal number of positive and negative instances, randomly sorted. 9 parts were used to train, while the last was used to test, rotating through each possible 9:1 combination and averaging the accuracies. Both a linear model and a 5-degree polynomial model were used for training and testing.

RESULTS

**Validation Accuracy**

Linear: 60.18%

Polynomial: 66.99%

**Training Accuracy**

Linear: 60.46%

Polynomial: 62.88%

**Testing Accuracy**

Linear: 59.80%

Polynomial: 61.41%

OBSERVATIONS

The overall accuracy between cross-validation and straight-up training on the full training set really didn’t appear to differ much when using the linear model. While the accuracy was better with cross-validation than the test accuracy, it was only minimally so. In fact, the full-data training performs better on the training set than the cross-validation.

The variance with the polynomial cross-validation model was more visible - over 4% better accuracy than that of the full training classifying the test set. Cross-validation actually proved a worthwhile investment of time; this was clearly the more accurate model. However, all results were still well under 70%, so there is still much room for this to improve.

ADABOOST

To script the Adaboost algorithm, I used Perl, from which SVM-Light functions were called. 10 iterations of boosting were implemented, calculating the error, alpha, and weight changes. I used a second array to hold each weight’s “barrier” so that each slice was represented as a range somewhere between 0 and 1. Therefore, when a random number was chosen for the roulette selection, it was more likely to fall within the larger slices, thus focusing more on these.

I found that boosting actually produced lower accuracy rates - about 61.02% - than cross-validation. However, this is still improved over the original accuracy rates, ranging between 49% and 52%.

Increasing the number of iterations didn’t appear to help much. 20 iterations still yielded an ensemble accuracy of 61.8%. These boosting iterations also took a significant amount of time; 20 iterations took about an hour to fully execute. With such minimal advantage, it hardly seemed worthwhile.

CONCLUSION

Cross-validation had a much greater impact on classification accuracies than Adaboost. I believe this has to do with the initial accuracy of classification. As the base algorithm sits right around 50% (at or below random), it is not surprising that Adaboost would be relatively ineffective, as it best improves accuracy when augmenting a classification with better-than-random performance. I might be curious to see if the two techniques could be combined, using the output data from cross-validation to feed into Adaboost. This could potentially produce better classification. However, comparing each independently, it is clear that cross-validation is far more effective.