Lab 3 Report

Ben Fu
EE 385T Intro to Machine Learning

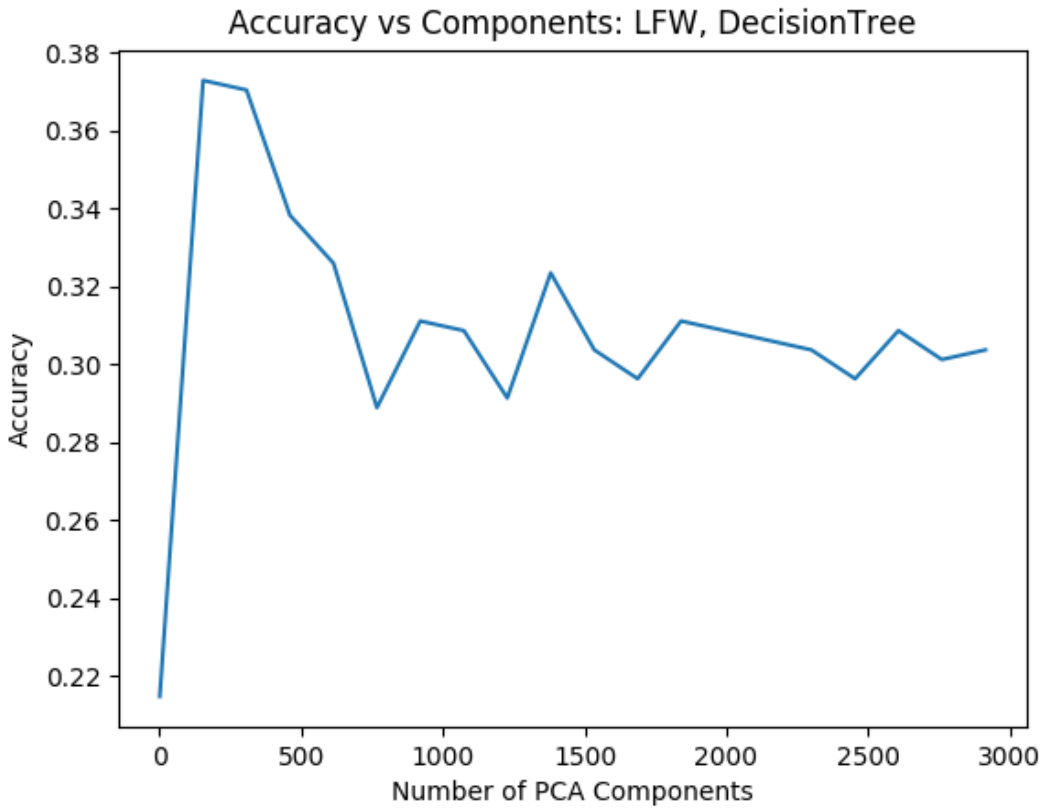
Code

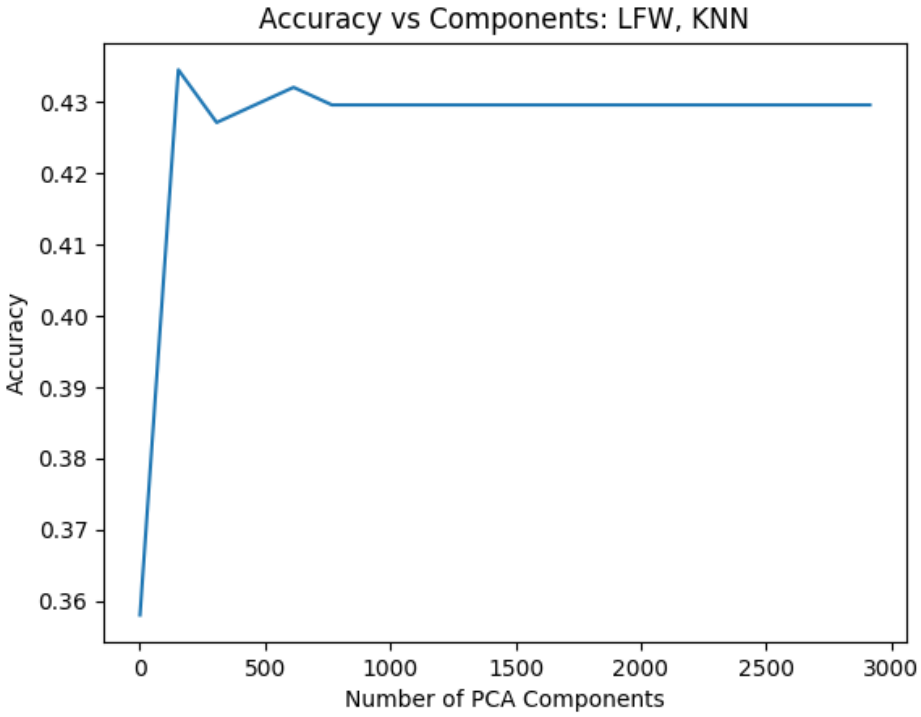
https://github.com/bennycooly/INF_385T_Intro_To_ML/blob/master/labs/lab3

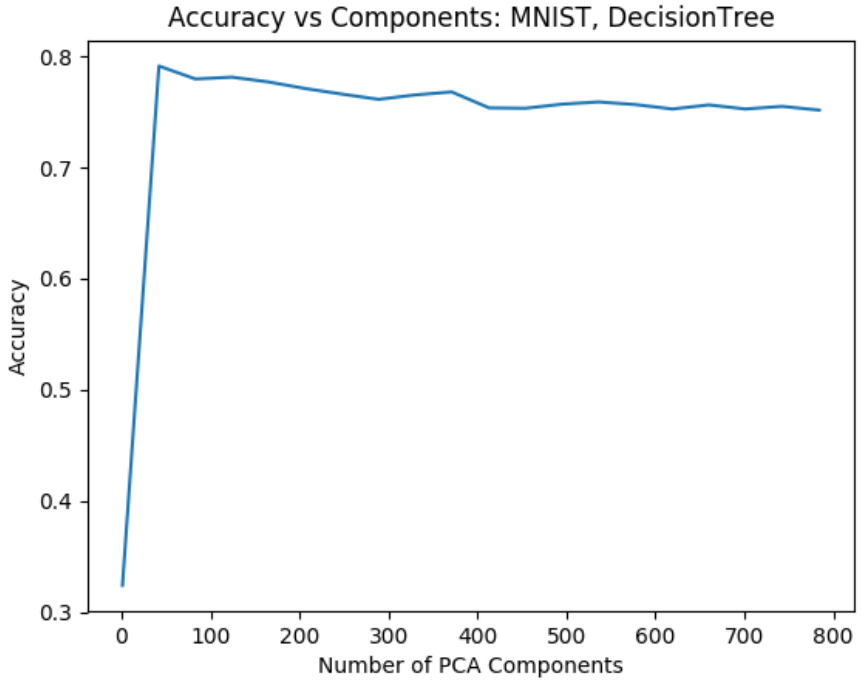
Questions

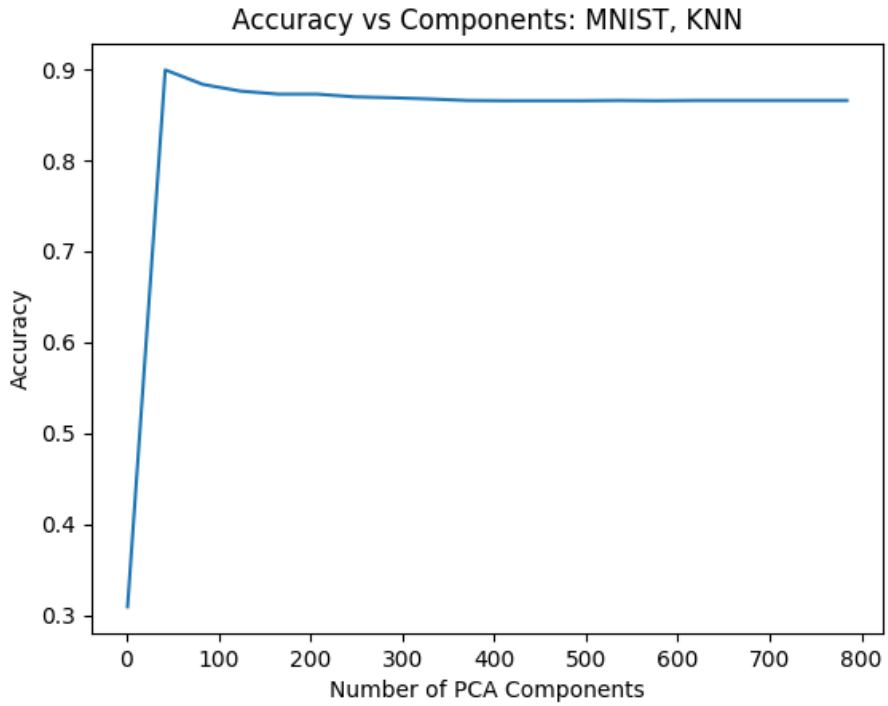
1. Image Classification with Dimensionality Reduction
    c. Accuracy Plots

Note: I distributed 20 evenly spaced components ranging from 1 to the actual number of features.




Accuracy vs Components: LFW, KNN
Accuracy
0.43
0.42
0.41
0.40
0.39
0.38
0.37
0.36
0
500
1000
1500
2000
2500
3000
Number of PCA Components


Accuracy vs Components: MNIST, DecisionTree
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0
100
200
300
400
500
600
700
800
Number of PCA Components



d. Discussion

Probably the most surprising result was the fact that the accuracy for face recognition was low even without any dimensionality reduction. The highest accuracy I found was around 44% when training with all of the features. However, even though the accuracy ceiling was low, the pattern with regards to the number of PCA components and the resulting accuracy was consistent with the MNIST dataset as well as the Iris and Breast Cancer datasets (included in the Github repo).

The plots show that when using a very low number of components, the accuracy is very poor. However, the accuracy rises steadily until it hits a ceiling (at about the 8% mark for LFW and the 10% mark for MNIST). After the ceiling, the accuracy slowly dips and becomes steady until the number of components reaches the full feature set.

This pattern makes sense because at low numbers of components, the model is underfitting by a lot, and the model's accuracy sharply rises in the first 10% of components until it hits a ceiling. This high point is where the model is not suffering from overfitting or underfitting. After this ceiling, the model begins to overfit and the accuracy dips a small amount (because the effects of overfitting for the remaining 90% are not as severe as those of underfitting within the first 10%). This result can be generalized to most classification problems.

2. Ensemble Learning

Dataset: Breast Cancer (from Sklearn)

c. Mean Accuracies

Model	Mean Accuracy
KNN	0.9298
Naïve Bayes	0.9387
Decision Tree	0.9106
Ensemble	0.9528
Bagging (KNN)	0.9281
Boosting (Decision Tree)	0.9754

Full output:

```
Running question 2
Loading dataset...
Mean accuracy (KNN): 0.929842926281
Mean accuracy (Naive Bayes): 0.938679673321
Mean accuracy (Decision Tree): 0.910637585343
Mean accuracy (Ensemble): 0.952778497969
Mean accuracy (Bagging, KNN): 0.928056131709
Mean accuracy (Adabooster, Decision Tree): 0.975434275344
```

d. Discussion

These results were pretty interesting. I chose to use the breast cancer dataset because it has a relatively high classification accuracy for all of the models we have discussed in class. For the single classifiers, the order of performance from highest to lowest mean accuracy was Naïve Bayes, KNN, and Decision Tree. This is pretty interesting because I did not expect the Naïve Bayes model to perform the best…however, this probably would have changed if I optimized the hyperparameters for the other models.

For the majority voting model, it is interesting that the accuracy is actually higher than any of the single classifiers. I thought that this was strange at first, but it makes sense because I did not randomize the state and the accuracy is based on 10-fold cross-validation.

The bagging model used the KNN classifier, and the mean accuracy was actually slightly lower than that of the single KNN classifier. This also makes sense because using a subset of the data may actually cause each individual subset accuracy to be lower than if the model was trained with the full dataset, so it is definitely possible that no individual accuracy score reached the threshold of the single classifier.

Lastly, the boosting model by default uses the Decision Tree classifier, and the resulting mean accuracy is much higher than that of the individual classifier (~6% difference). This is expected because the boosting model looks at the misclassified data and updates the weights to improve the error rate. With more iterations, even higher accuracy could be achieved.