CS 7641: Assignment #1
Supervised Learning

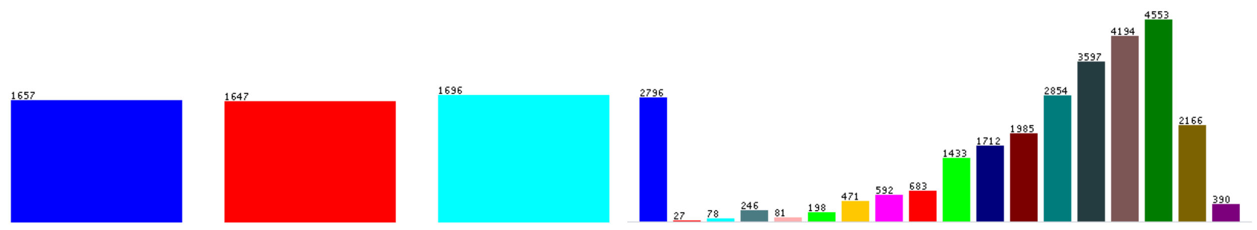
Due on Sunday, September 25, 2016

Dr. Charles Isbell

Ryan Chow

Introduction

This report explores techniques in supervised learning by comparing the performance of five popular algorithms against two datasets. The first dataset is Waveform Database Generator and contains 5,000 instances. Each instance represents a waveform consisting 21 attributes of continuous values between 0 and 6. The goal of this dataset is to classify each waveform into 3 base wave categories. The second dataset is Chess King-Rook vs. King and contains approximately 28,000 instances. Each instance represents a chess board status consisting 6 nominal attributes (the column and row of three chess pieces). The goal of this dataset is to classify each chess board status into optimal depth-of-win for the King-Rook, ranging from zero to sixteen moves or a draw. Both datasets are available from the UCI Machine Learning Repository online. A histogram of each dataset's output variable is shown in Figure 1 below.



(a) Waveform dataset (5,000 instances) (b) Chess dataset (28,056 instances)

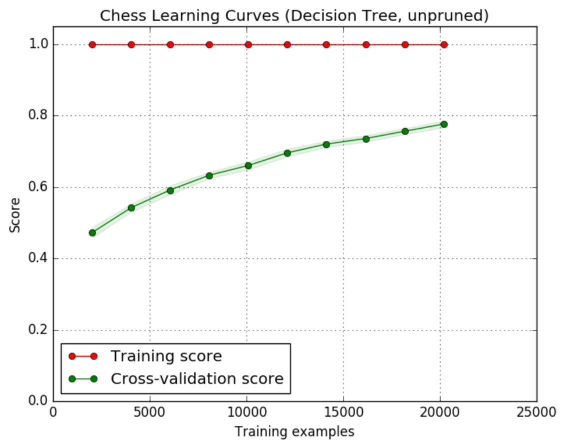
Figure 1: Classification histograms

The chess dataset was interesting because of the recent news about Google's artificial intelligence system that beat the world champion of Go. Both games have complex move sequences and require long-term planning. Go, however, has many more degrees of freedom of gameplay than chess. The chess dataset, limited to a scope of three pieces, would be an interesting supervised learning problem and provide insight into how Google trained the Go algorithm. The waveform dataset would also be interesting because it demonstrates how machine learning applies to natural phenomena such as light or sound waves. Even if the dataset was artificially generated, the machine learning techniques can show how data in nature can be analyzed.

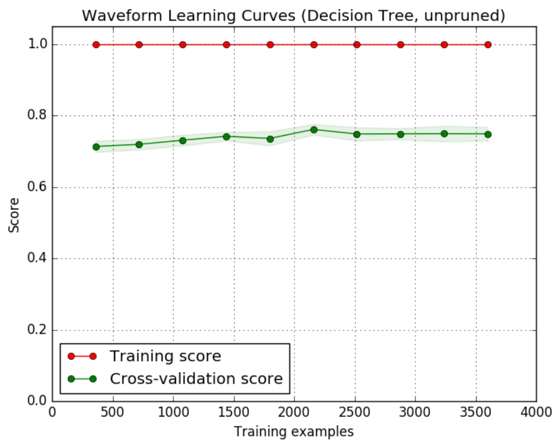
Together, the datasets provide interesting classification problems. They are almost dichotomous in how the data are represented; the Chess dataset contains only nominal attributes, which are used to predict 18 possible classifications. The Waveform dataset contains only continuous attributes, which are used to predict just 3 possible classifications. In addition, the Chess dataset contains nearly five times as many instances. The underlying structure of these datasets illuminate many differences in performance between algorithms. A final analysis at the end of this report summarizes the lessons learned and key factors affecting the performance of algorithms for each dataset.

Decision Trees

All decision trees in this section use the Gini impurity to split nodes. The unpruned decision tree learning curves below (Figure 2) illuminate clear variance issues. The Chess training set maintains nearly perfect accuracy as more examples are added, while validation set accuracy rapidly increases without ever plateauing. The Chess dataset suffers from a high variance issue, and adding additional training examples should help to increase validation set accuracy. The Waveform dataset shows no change in accuracy for training or validation sets, showing poor ability to generalize beyond the training set. Additional training examples will not improve accuracy performance. Because the Waveform attributes are continuous, the algorithm may be creating too many nodes for each bucket of continuous data, causing overfitting on the training data.



(a) Chess dataset

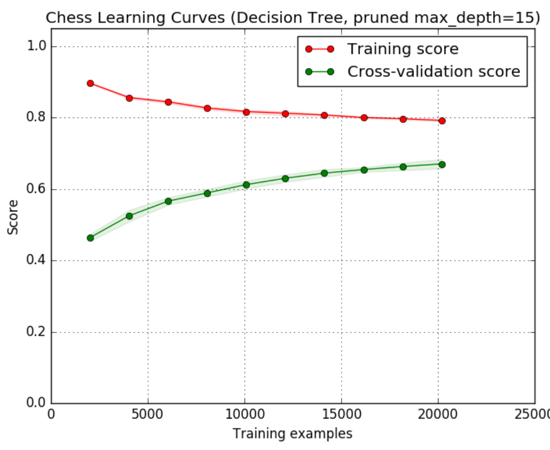


(b) Waveform dataset

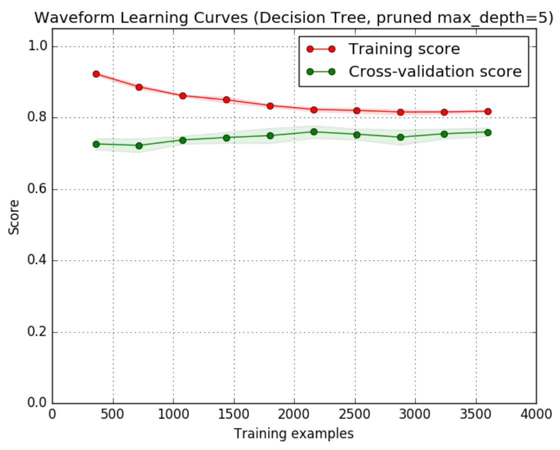
Figure 2: Decision tree learning curves (unpruned)

As expected, pruning the decision trees as shown on the next page (Figure 3) reduces variance for both datasets compared. Pruning was performed by limiting the maximum tree depth. The Chess dataset performs poorer on the validation set compared to unpruned because a bias is introduced by limiting the complexity of the model. The Waveform validation set performs similarly to unpruned, but the pruned model still suffers from a bias issue. I tried pruning using two other methods - changing the min number of samples required to split a node and min samples required for a leaf, but max tree depth provided the lowest overall error out of my experiments. Scikit-learn does not support the more popular post-pruning methods, which was discussed in the lecture videos. That method of pruning may help to improve the underlying model and reduce bias.

Decision Trees continued on next page. . .



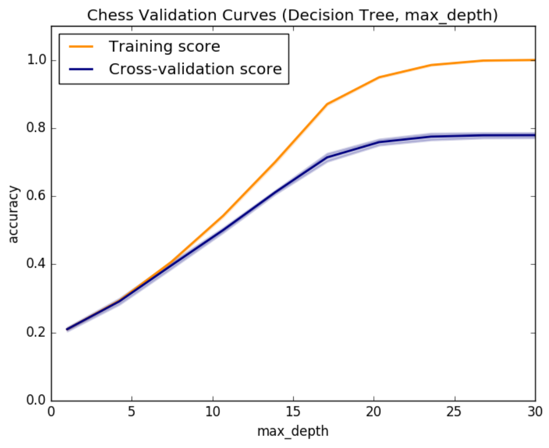
(a) Chess dataset



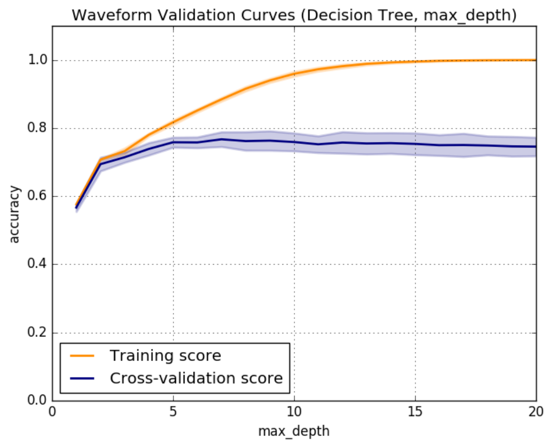
(b) Waveform dataset

Figure 3: Decision tree learning curves (pruned)

Looking at the validation curves of maximum tree depth below (Figure 4), it is not surprising that pruning the decision trees reduces generalization error up until a certain point (while maintaining same validation set accuracy). At max depth=15 and max depth=5 for the Chess and Waveform datasets, respectively, both the validation and test set accuracies begin to plummet. Pruning the decision trees help to eliminate overfitting, but too much pruning results in losing the model complexity required to accurately classify the data. I used these pruning parameters to generate the learning curves above.



(a) Chess dataset



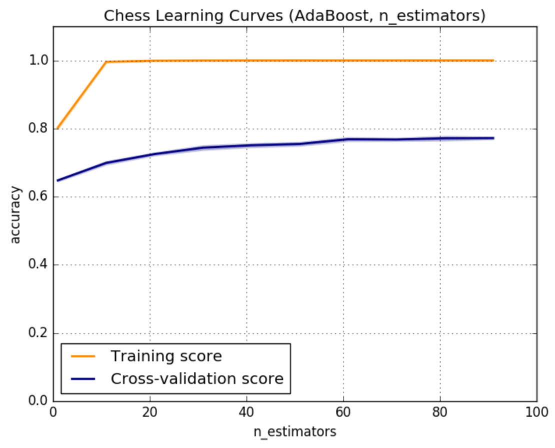
(b) Waveform dataset

Figure 4: Decision tree validation curves

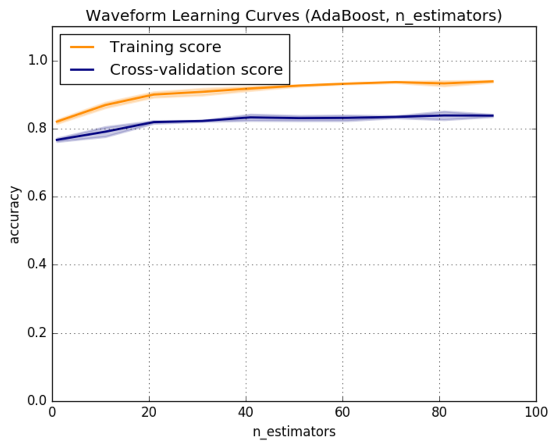
Note: x-axis scale ranges from the left of the graph where depth=1 (most pruning) to the right of the graph where depth=30 (least pruning). Pruning amount increases as the curve is followed from right to left.

Boosting

Pruned decision trees from the previous section were used as the base estimators for AdaBoost in this section. Boosting increased testing accuracy for both datasets, as shown in Figure 5 below. However, it is interesting to note that the boosted Waveform validation accuracy surpassed the validation accuracies for both pruned and unpruned examples in the previous section. I think boosting benefitted the Waveform dataset more because given the training data, boosting created a better underlying model that minimized overfitting compared to a single estimator. On the other hand, the Chess dataset did not suffer nearly as much from overfitting and the model simply needed more training examples to improve performance on test data.



(a) Chess dataset

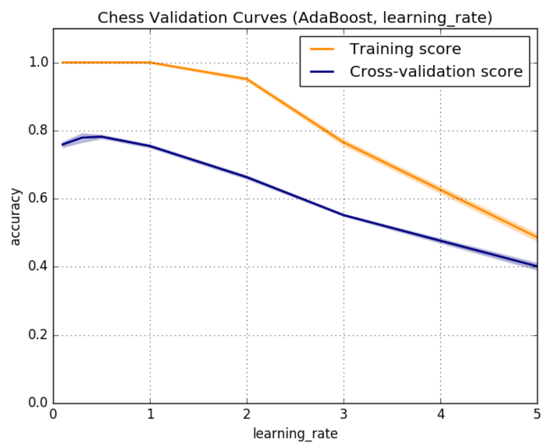


(b) Waveform dataset

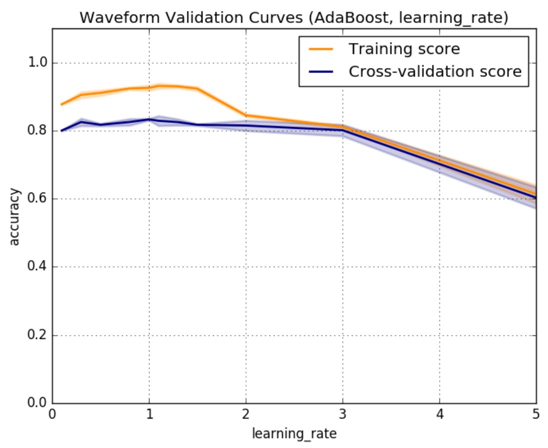
Figure 5: AdaBoost learning curves (pruned decision tree base estimator)

The learning curve for learning rate in AdaBoost is shown on the next page (Figure 6).The AdaBoost learning rate parameter changes the contribution of each classifier to the overall model. Generally, lowering the learning rate reduces overfitting because the weight of classifiers that aren't optimal are lowered. However, it is very interesting that the Waveform dataset seems more robust (its accuracy does not drop off as much) to larger learning rates compared to the Chess dataset. Theoretically, the underlying structure of Waveform data is much more uniform than Chess data because waves are a natural phenomena while chess does not seem to follow any patterns. For AdaBoost, this means that the Chess dataset quickly declines in accuracy as learning rate is increased and higher weights are placed on classifiers that are not optimal. The Waveform dataset has fewer of these high-leverage classifiers and is therefore more robust to overfitting.

Boosting continued on next page...



(a) Chess dataset

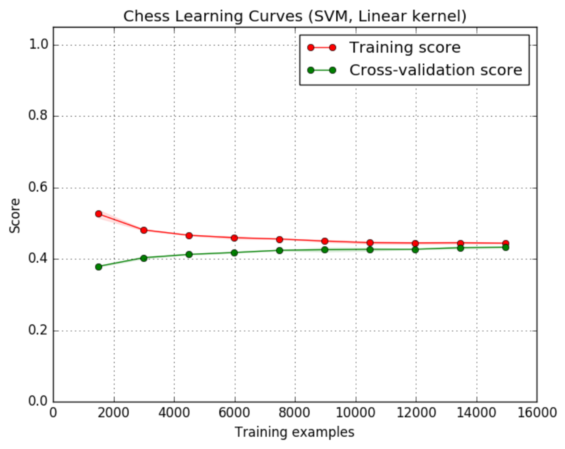


(b) Waveform dataset

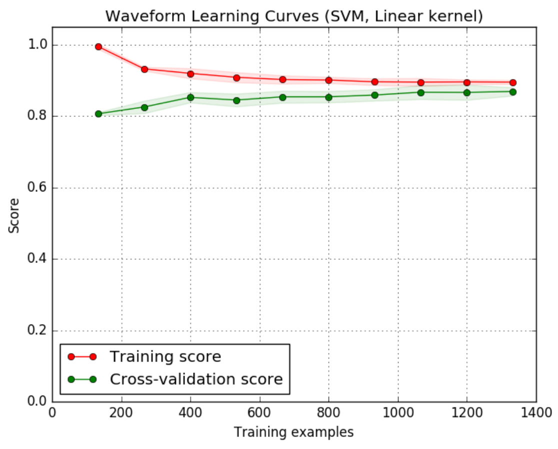
Figure 6: Adaboost validation curves (pruned decision tree base estimator)

SVM

My previous theory about the Waveform data being more 'uniform' compared to the seemingly chaotic Chess data is further validated with the Support Vector Machine learning curves shown below (Figure 7). The Waveform dataset performed very well using SVM with a linear kernel. This means that the algorithm was able to find an adequate linear boundary between classes, leading to about 90 percent classification accuracy. The Chess dataset, however, is not nearly as linearly separable and the training/validation scores converge at around 40 percent accuracy.



(a) Chess dataset

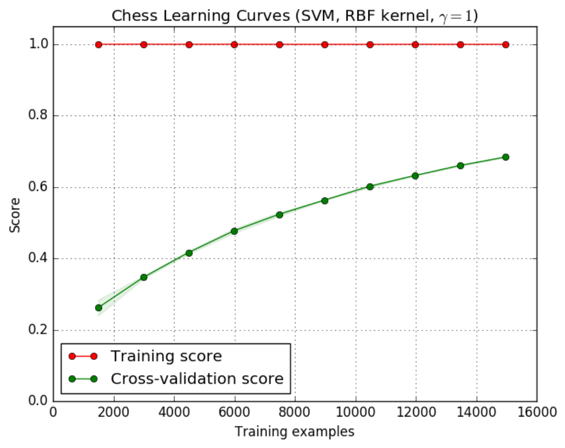


(b) Waveform dataset

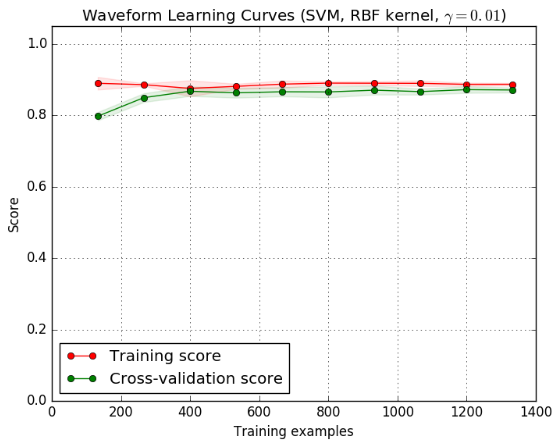
Figure 7: Support vector machine learning curves (linear kernel)

SVM continued on next page...

Figure 8 below shows each dataset with Support Vector Machine using the radial basis function (RBF) kernel. The Chess dataset performs significantly better, which means that RBF was able to convert the nonlinear dataset to a higher dimension and find a linearly separable boundary. The validation test result increased by approximately 30 percent, and the learning curve shows even more improvement could be made if additional training samples were available. However, the Waveform dataset using RBF kernel fails to improve over the linear kernel. This is because the linear kernel was already able to find a good support vector in the original dataset, and converting the data to a higher dimension did not result in a better linear boundary.



(a) Chess dataset



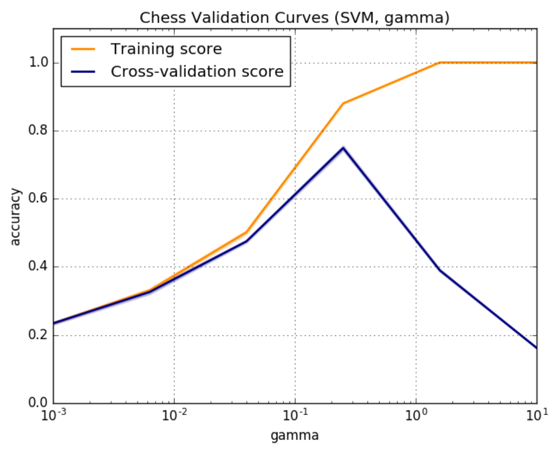
(b) Waveform dataset

Figure 8: Support vector machine learning curves (RBF kernel)

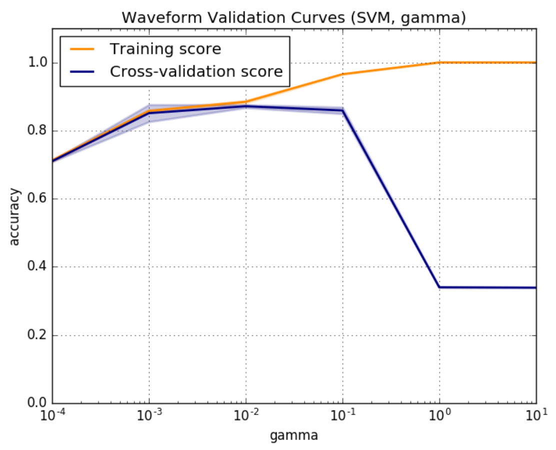
The validation curve for SVM using RBF kernel and tuning the gamma parameter is shown on the next page (Figure 9). The gamma parameter is significant in SVM and has a large effect on performance because it relates inversely to the width of each bell-shaped surface centered at each support vector. As shown in the figures below, low values of gamma underfit the data. At approximately gamma=0.1 for both datasets, however, the model begins to overfit the data and training accuracy continues to increase while testing accuracy plunges. This means that high values of gamma cause overfitting. This comparison is interesting because the Waveform validation curve plateaus from gamma=0.0001 to gamma=0.1 with accuracy staying above 75 percent, while the Chess dataset peaks at a single gamma value with an accuracy of about 75 percent. This leads me to believe that the Waveform dataset is more robust to underfitting because the underlying data has a larger separation boundary between classes compared to the Chess data.

In addition, I tried to tune the SVM model using polynomial kernels with varying polynomial degrees with low success compared to linear kernel for Waveform dataset or RBF kernel for Chess dataset. For the Chess dataset, this may be because the complexity of RBF kernel can increase exponentially with the dataset complexity, while a polynomial kernel is limited to a fixed size. The RBF gaussian kernel also smoothes the highly complex Chess data. However, I did only experiment with low-number polynomial degrees with the polynomial kernel due to CPU time.

SVM continued on next page...



(a) Chess dataset

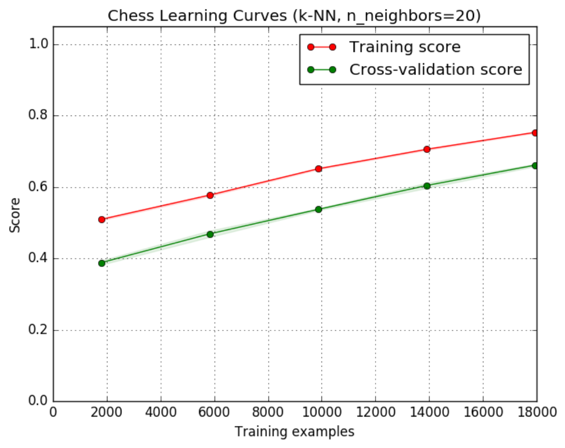


(b) Waveform dataset

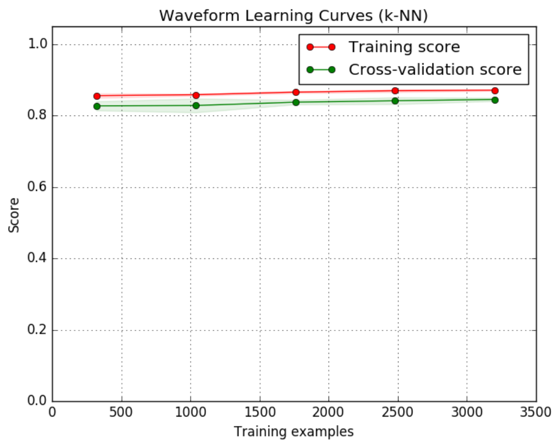
Figure 9: Support vector machine validation curves (RBF kernel)

kNN

KNN for the Chess dataset also shows that additional training data would help improve accuracy scores (Figure 10). It is clear that as training examples increase, the accuracy also increases. However, the Waveform data is peculiar because that is not the case. The Waveform dataset reaches approximately 80 percent peak accuracy at 300 training samples and stays at the same accuracy even as 3,000 additional training samples are added. This means that the solution KNN found using 10 percent of the available data was representative of the whole dataset. This again shows that the underlying Waveform data groups are highly separated from each other. The Chess data continues to learn as more data is added because of the highly complex underlying data structure.



(a) Chess dataset

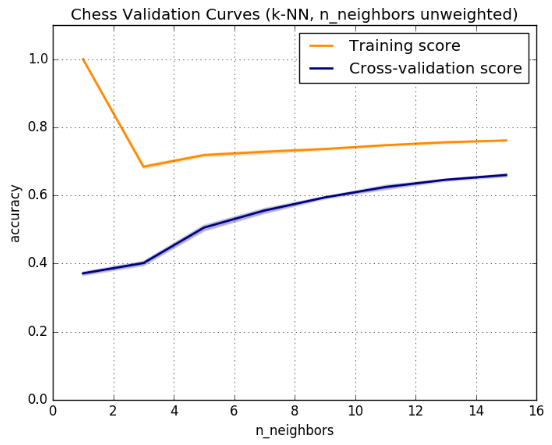


(b) Waveform dataset

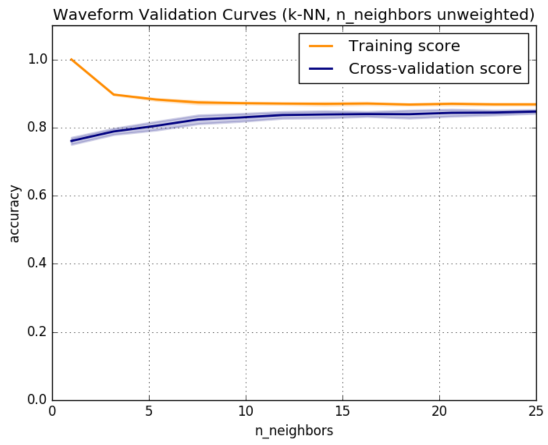
Figure 10: K-nearest neighbors learning curves

kNN continued on next page. . .

I chose to use validation curves to experiment with number of nearest neighbors in K-nearest neighbors because, as the name suggests, it highly influences algorithm performance. This plot is shown below in Figure 11. The Chess dataset showed a significant (almost 30 percent) improvement in accuracy from 1 to 15 nearest neighbors. This could mean that the boundaries between classes are very tight and highly nonlinear. This evidence of complex boundaries has repeated itself from other algorithms analyzed in this report such as SVM in the previous section. The Waveform dataset, however, has proven to have a large separation between classes. Even at 1 nearest neighbor, KNN performs very well on the Waveform dataset and accuracy only increases a couple percent when 25 nearest neighbors are used. Weighted KNN methods were also used, but showed no significant improvement over the unweighted examples shown above.



(a) Chess dataset

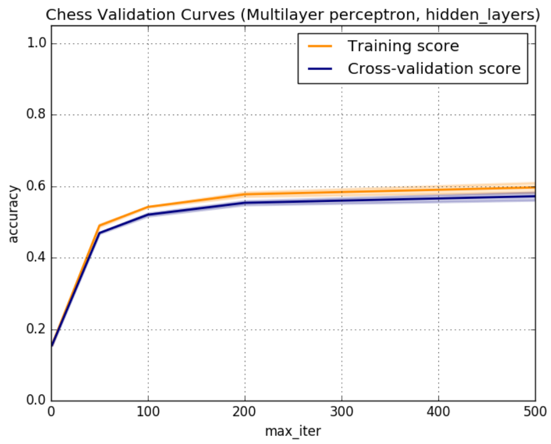


(b) Waveform dataset

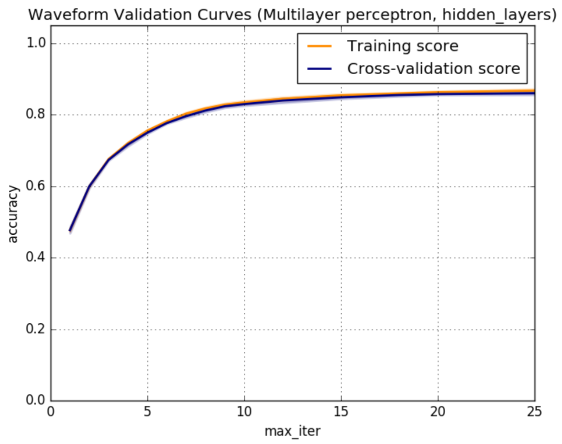
Figure 11: K-nearest neighbors validation curves

Neural Networks

For neural networks, I used scikit-learn's development version of Multilayer Perceptron. This experiment yielded interesting results shown below in Figure 12. The Chess dataset took approximately 200 epochs to reach its maximum accuracy of 60 percent, while the Waveform dataset took approximately 15 epochs to reach its maximum accuracy of 85 percent. The Waveform dataset settled sooner most likely because the gradience descent converged far quicker for 3 output nodes compared to 18 for the Chess dataset. In addition, the 18 output nodes required far more hidden layers to train as shown below in the validation curve of Figure 13.



(a) Chess dataset

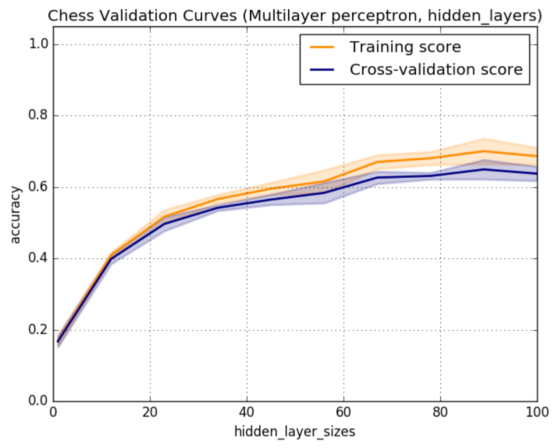


(b) Waveform dataset

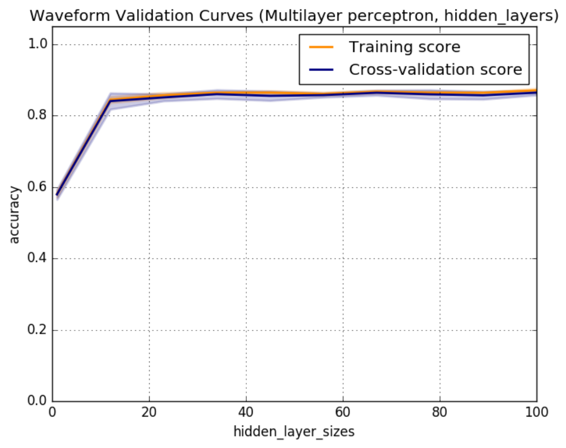
Figure 12: Multilayer perceptron learning curves

Neural Networks continued on next page...

I also ran these experiments by varying the number of hidden layers in the neural network. This further illuminated the complexity differences between the underlying datasets. The Waveform dataset showed an accuracy score plateau after approximately 15 hidden layers, while the Chess dataset continues to improve in training/test accuracy even after 100 hidden layers. I also experimented with various solver functions, and Scikit-learn's default 'adam' optimized gradient descent performed best and converged quickest. I could have experimented with many more model complexity parameters such as adjusting learning rate, momentum and activation functions, but the scope was too large for this project.



(a) Chess dataset



(b) Waveform dataset

Figure 13: Multilayer perceptron validation curves

Final Thoughts

Waveform Dataset: The Waveform dataset presented a recurring theme through almost all the experiments run. This theme was a maximum achieved value for training/testing accuracy equal to approximately 85 percent. Evident in k-NN, neural networks, SVM, and AdaBoost (with single decision tree being slightly less), all algorithms performed similarly after parameter tuning and adjusting model complexity. This is very interesting, as working with each algorithm one-by-one showed me how the intricacies of each technique were so unique. However, at the end, all the algorithms performed within plus or minus five percent. This was extremely interesting to me to realize in retrospect, but is reasoned throughout this report because the underlying data was found to be easily separable.

Chess Dataset: The Chess dataset testing set varied between 60 percent and 80 percent accuracy for all algorithms. This is not surprising, considering the underlying structure of the chess data is so much more complex than the Waveform dataset. It was interesting to see how each algorithm parameter adjusts the model complexity to try to fit the data. It was evident that this dataset lacked sufficient training examples to maximize performance, but there was enough data to illuminate difference between how each algorithm worked. For all algorithms I used k-fold cross validation between 3 to 10 folds, depending on reasonable cpu runtimes for each experiment. The cross validation definitely helped to reduce variance in the datasets, which was especially evident with this dataset since it was lacking training data. I believe that given enough training data (likely 50,000 or more instances) the algorithms would converge on a solution, although training time would be very long.

CPU Time: The two algorithms that took the longest time were eager-learning SVM and Multilayer Perceptrons. This was the limiting factor in finding the absolute best-performing algorithm. With infinite computing power, I could just run a grid search cross validation to find the best model parameters for my datasets. However, given the time and computer power I had, I think I found model parameters for each algorithm that allowed me to sufficiently compare the intricacies and performance of each technique. Default scikit-learn values were used for any algorithm parameters not mentioned in this report.

Scoring: Classification accuracy seemed to be a valid scoring assessment of the machine learning techniques used in this assignment. The baseline score was approximately 16 percent for the Chess dataset and 33 percent for the Waveform dataset. This baseline score is simply the percentage of the most-represented outcome class. All algorithms performed significantly above this baseline; however, if my dataset output was a skewed binary classifier with a 95 percent / 5 percent distribution for example, my baseline score would be 95 percent and I would consider using other scoring metrics to evaluate the algorithms. Although, I could have used confusion matrices to see if the incorrect classifications were within plus or minus 1 of the true value. This metric would have been extremely useful for the Chess dataset where the algorithm tried to predict the number of moves until checkmate, and less useful for the Waveform dataset where there were only 3 classification categories.