CS7642 Assignment 1: Supervised Learning

Xiaoming Su

Introduction

This assignment aims to perform analysis of five well-known supervised machine learning algorithms on two classification problems. The algorithms studied are: decision tree, boosting, neural network (ANN), K-nearest neighbors (KNN) and support vector machine (SVM).

The strength and weakness of different algorithms were analyzed and the parameters for each problem were optimized as well.

All the algorithms are implemented under python 3.5 environment using sklearn and keras.

There are 3 sections in the report:

1. Interesting datasets
2. Algorithms Analysis for Dataset 1 and Dataset 2
3. Comparison and Conclusion.

1. Interesting Datasets

Dataset 1: Wine Quality Data Set

The objective is to determine the quality of red wine based on 11 features with no missing values. The quality of the wine is rated between 0 to 10. There are 1599 instances.

The dataset is interesting because it can be considered as classification or regression tasks. It have moderate amount of instances and attributes, allowing for fast iteration while testing various algorithms and compare their strength and weakness. In order to compare with dataset 2, the dataset was viewed as a classification problem.

Recourse:

https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Dataset 2: Spambase Data Set

The objective is to identify whether an email is a spam based on 57 features generated by analyzing real world spam and non-spam emails. All the attributes are real numbers with some missing values. The spambase dataset has 4601 instances.

This dataset is interesting based on several reasons. First, this is a classical binary classification problem. It helps to evaluate different algorithms' performance in solving this typical problem. Second, it has some missing values, thus it has the potential to evaluate the tolerance of different algorithms for missing values, which is very common in real world dataset. What's more, compared with dataset 1, it has much more attributes and instances. Thus, it enables to compare the dimensionality and scalability influence on different supervised learning algorithms. In the end, identifying large amount of spam emails in an efficient and robust way is a very import problem that can make our daily life much easier.

Resource:

https://archive.ics.uci.edu/ml/datasets/Spambase

The 2 datasets are both classification problems and they have several differences that can help to compare the five algorithms. They are different in particular ways, e.g. number of attributes, existence of missing values, binary/multi classification, they are a good pair to compensate each other for algorithm study.

2. Algorithms Analysis for Dataset 1 Wine Quality and Dataset 2 Spambase

2.1 Decision Tree

The decision tree algorithm was implemented by using sklearn.tree.DecisionTreeClassifier. Pruning was controlled by setting the max depth parameter. The decision tree is a very fast algorithm, for this wine dataset, it only takes about 200 secs to run all the analysis. Decision tree is an eager algorithms, which means it takes more time on training than on testing. Compared with other methods, it's really efficient.

2.1.1 Wine

Figure 1. DT Error VS Tree Depth

Figure 1 illustrates the relationship between error (1-accuracy) and tree depth. When the tree depth is below 10, all the error decrease. When the depth >10, overfitting occurs (not very obvious). Pruning (reducing tree depth) is widely used to prevent overfitting in decision tree algorithm.

Figure 2. DT Error VS Training Size

Figure 2 shows the relationship between the error and training size. The depth was set at the optimal value for the analysis. As the training size increases (< 900), all the errors decrease with the largest decrease of training error. When the training size is really large, there's little increase of cross validation and testing error.

2.1.2 Spam

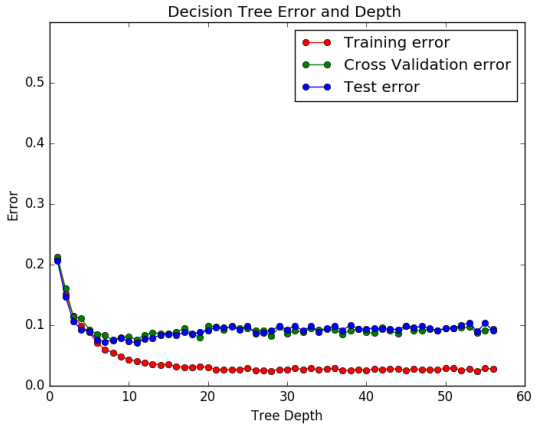


Figure 3. DT Error VS Tree Depth

Figure 3 illustrates the relationship between error and tree depth. Overfitting occurs when the tree depth is larger than 8.

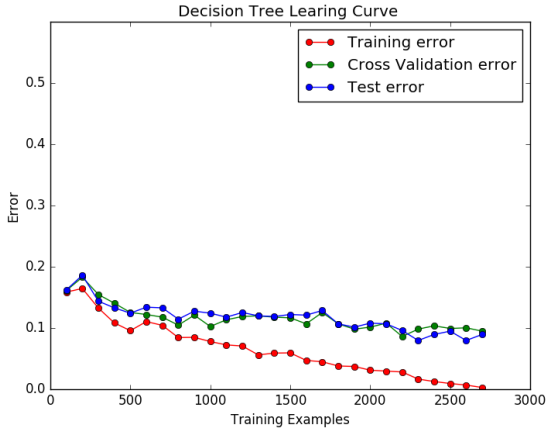


Figure 4. DT Error VS Training Size

Figure 4 shows the relationship between the error and training size. The depth was set at the optimal value for the analysis. As the training size (> 200) decreases, all the errors decrease with the largest decrease of training error.

The spam dataset, compared with wine dataset, has much more attributes, it is more influenced by pruning based on the figures about error vs tree depth. As the dimensionality of the dataset gets larger, it's important to using pruning to prevent overfitting.

2.2 Boosting

The wine and spambase datasets were analyzed using Adaboost algorithm with the same decision tree

algorithm used in the 2.3. The decision tree is used as weak learner in boosting. Besides pruning, boosting is another method that is used to prevent overfitting. The runtime linearly increases with the size of the training data. The influences of training size, learning rate, tree depth and estimators on error (error = 1-accuracy) were analyzed.

2.2.1 Wine

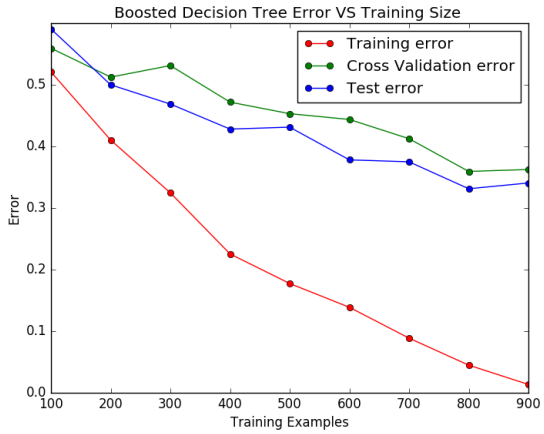


Figure 5. Boosted Error VS Training Size

Figure 5 shows the relationship of error VS training size. All the errors decrease as the training sample size increases. Thus the performance of the model can be increased by increasing training size.

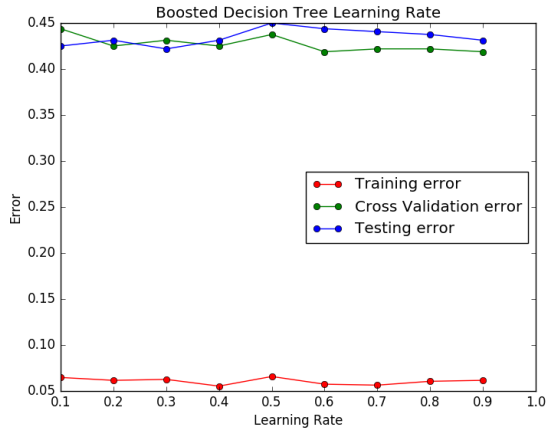


Figure 6. Boosted Error VS Learning Rate

Figure 6 shows the relationship between error and learning rate. Learning rate has very little influence on the training/cross validation/testing error. Unlike other algorithms, shrunken learning rates do not improve AdaBoost on this dataset. This result matches the result from the literature report [1].

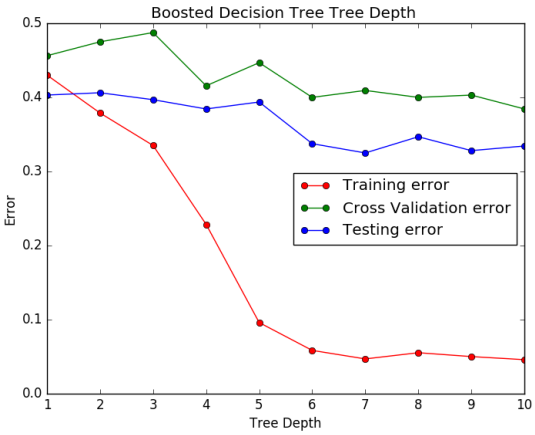


Figure 7. Boosted Error VS Tree Depth

Figure 7 shows the relationship between error and tree depth. Boosting can prevent overfitting.

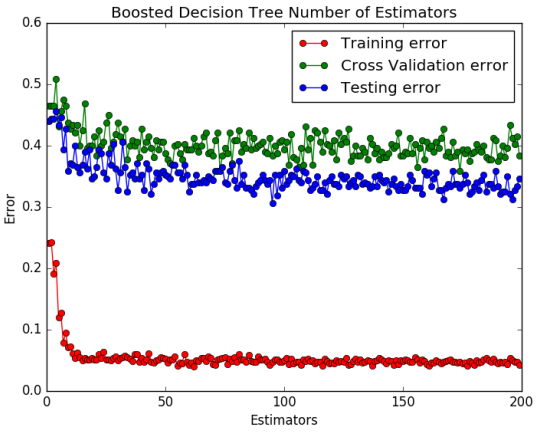


Figure 8. Boosted Error VS Estimators

Figure 8 shows the relationship between error and estimators. The number of weak learners is controlled by the parameter n_estimators. The error can be decreased by increasing the number of weak learners. However, when the number of the week learner is really large, there's no large increase of error. The optimal number of estimators can be found based on the figure.

By analyzing the wine data, Adaboost is quite robust to overfitting in dealing with data of relatively small dimensionality.

2.2.2 Spam

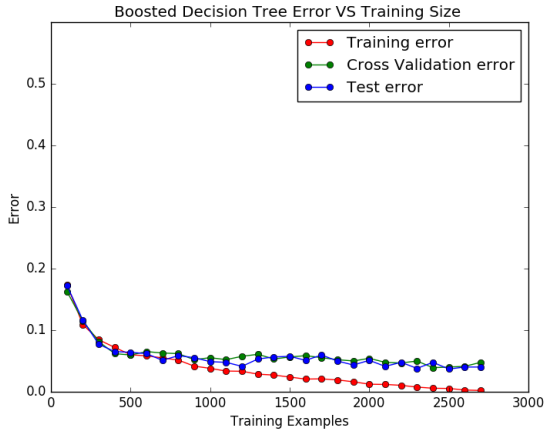


Figure 9. Boosted Error VS Training Size

Figure 9 shows the relationship of error VS training size. All the errors decrease as the training sample size increases. When the training size is really large, the drop rate of cross validation error and testing error slows down. The training error drops faster than the others.

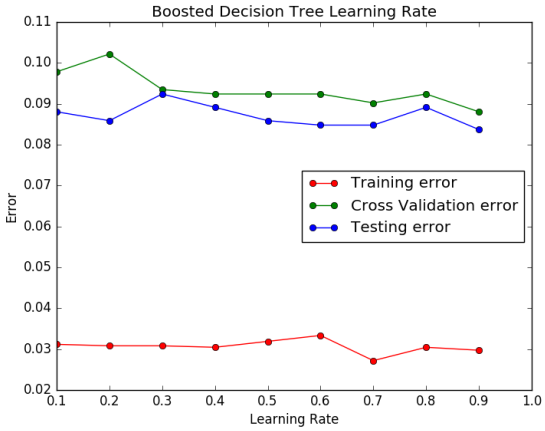


Figure 10. Boosted Error VS Learning Rate

Figure 10 shows the relationship between error and learning rate. Decreasing learning rate does not show obvious improvement for the algorithm. Thus, the reduced learning rates cannot be recommended for use with boosted decision trees on datasets similar to these datasets. The research from Forrest Daniel demonstrate this result [1].

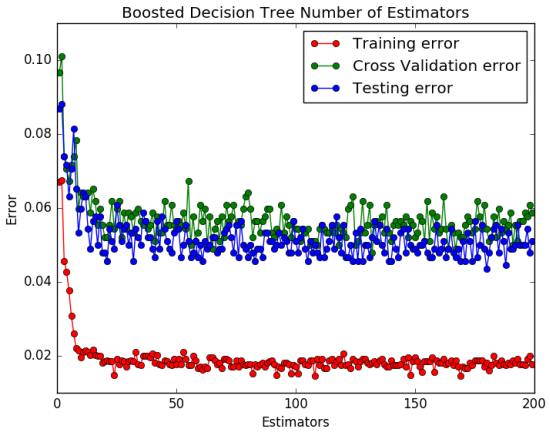


Figure 11. Boosted Error VS Estimators

Figure 11 shows the relationship between error and estimators. The error can be decreased by increasing the number of weak learners. When the number of the week learner is really large, there's no large increase of error. The optimal number of estimators can be found based on the figure.

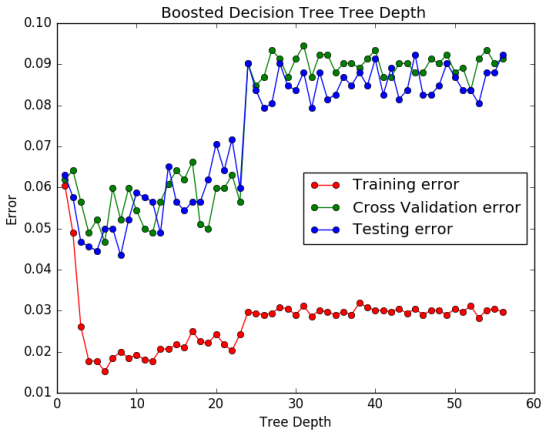


Figure 12. Boosted Error VS Tree Depth

Figure 12 shows the relationship between error and tree depth. However, when the tree depth is large, all the errors increase, though in a very small amount (the scale is small). This is not overfitting, but it's interesting to consider the overfitting problem of this algorithm.

Overall, AdaBoost is quite robust to overfitting. But it's also influenced by several main factors:

- The "strength" of the "weak" learners
- The noise level in the data. AdaBoost rarely suffers from overfitting problems in low noise data cases. However, studies with highly noisy patterns have clearly shown that overfitting can

occur [2].

- The dimensionality of the data. Overfitting is more likely to occur in high dimensional spaces ("the curse of dimensionality").

AdaBoost can suffer in these respects, as it is simply a linear combination of classifiers which themselves suffer from the problem.

2.3 ANN

Neural network is eager algorithm. The training time is really long while the testing is quite short. Overall, it is a relatively slow method with high epochs. The training time is proportional to the epochs.

2.3.1 Wine

For this dataset, the network topology was determined to be one-layer neural network. Different optimizers (adam, RSMprop and SGD) were evaluated for the best performance. For the SGD optimizer, different values of learning rate and momentum was chosen. The neural network are very sensitive to learning rate and momentum, the accuracy are highly influenced by these parameters (Adaboost is not sensitive to these parameters). If the learning rate was too high, the performance was not good. Among all the optimizers, RSMprop was found to have the highest accuracy for the problem.

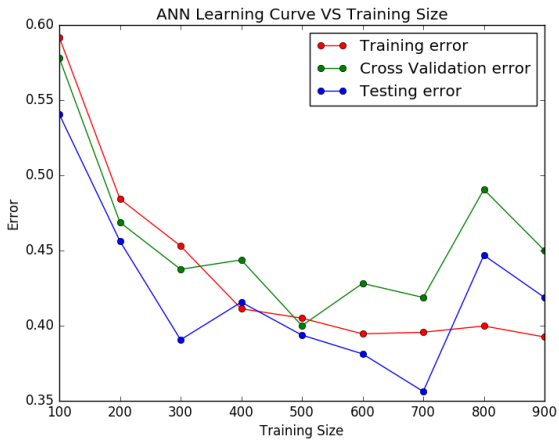


Figure 13. ANN Error VS Training Size

The Figure 13 shows the relationship between training size and error. The error was calculated by the equation error=1-accuracy. The training size decreases with the increase of training sample size. However, the testing error and cross validation error deceases at the beginning and increases a little bit when the training size was higher than 700.

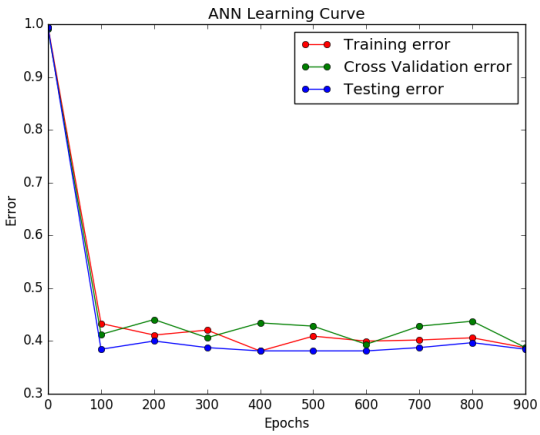


Figure 14. ANN Error VS Epochs

Figure 14 shows the relationship between epochs and error (error was calculated by 1-accuracy). The training error decreases really fast with the increase of epochs and then changes in a relatively small range when epochs are higher than 100. The cross validation and testing error show similar behavior. The overfitting when epochs are really large is not really obvious.

The wall clock time taken linearly increases with number of epochs.

Figure 15 and 16 are about the grid search for testing and cross validation errors of data1.

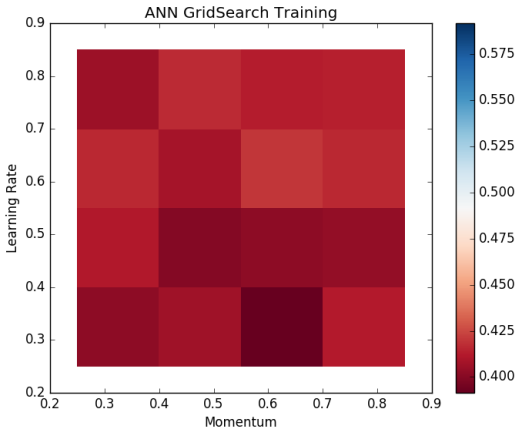


Figure 15. ANN Grid Search Training Set

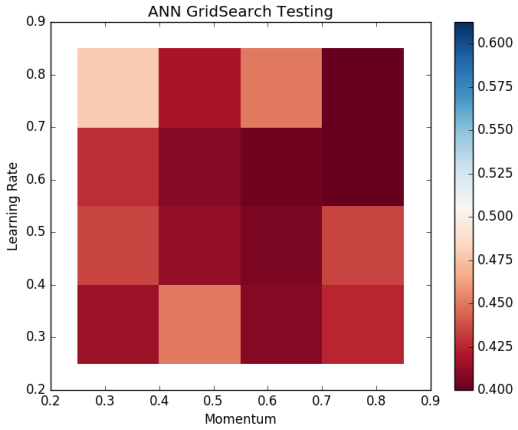


Figure 16. ANN Grid Search CV Set

2.3.2 Spam

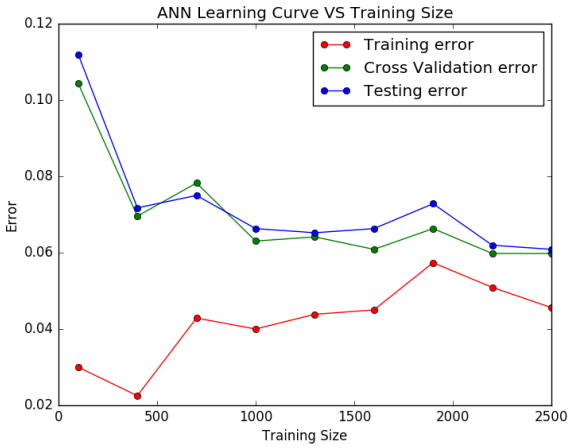


Figure 17. ANN Error VS Training Size

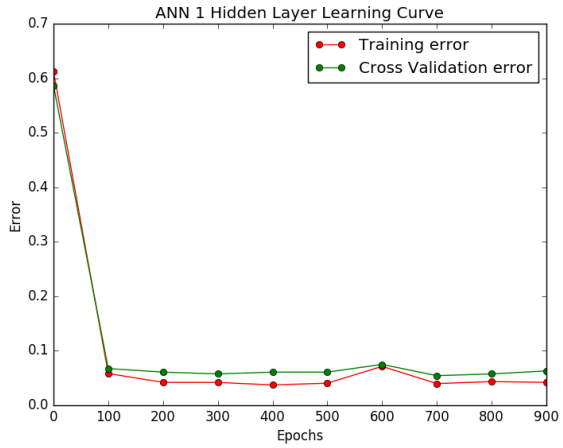


Figure 18. ANN Error VS Epochs

The Figure 17 shows the relationship between training size and error. The error was calculated by the equation error=1-accuracy. Compared with dataset1, the ANN shows far better performance on the spambase dataset. The training size shows opposite change direction compared with wine dataset, it increases with the increase of training sample size. This is not common, because the learning rate was set to be very small. The testing error and cross validation error decrease with the increase of training size.

Figure 18 shows the relationship between epochs and error. The training error and cross validation error drop really fast with the increase of epochs.

2.4 KNN

KNN is a lazy algorithm, it can be updated easily but needs to take more time on testing. For both of the 2 datasets, the KNN runtime was short comparing with the other algorithms.

For the KNN algorithms, two different weight function were used in analysis:

Distance: Weight points by the inverse of their distance. Closer neighbors of a query point will have a greater influence than neighbors which are further away.

Uniform: Uniform weights. All points in each neighborhood are weighted equally.

2.4.1 Wine

Distance:

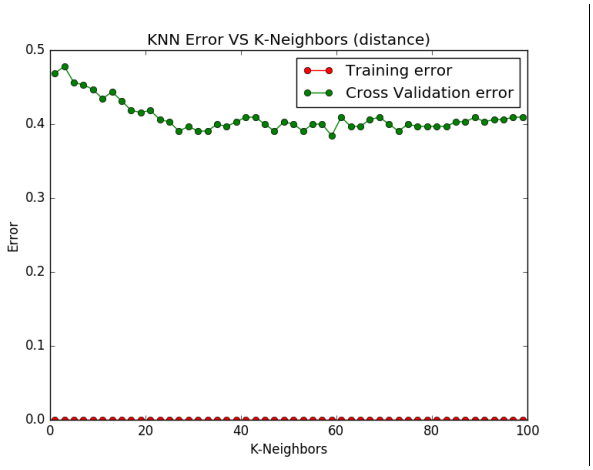


Figure 19. KNN Error VS K-Neighbors (Distance)

Figure 19 shows the relationship between error (error was calculated by 1-accury) and the number of neighbors. For the training dataset, the error are always zero. For cross validation error, it decreases as K increases. When K are approaching 100, the error

shows very slow increase. This makes sense, if the K is really large, the algorithm cannot model the local environment really well.

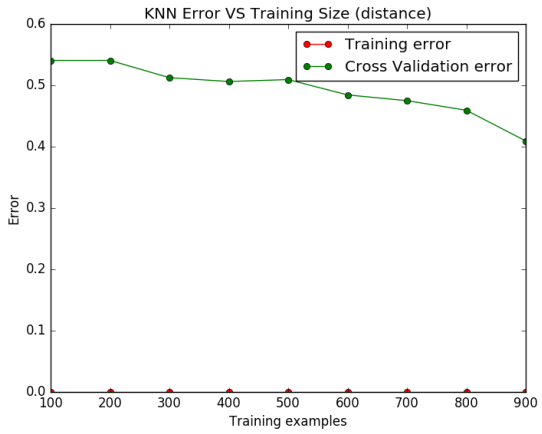


Figure 20. KNN Error VS Training Size (Distance)

Figure 20 shows the relationship between error and training sample size. Interestingly, the training error was 0 all the time. It's possible because it's a classification problem instead of a regression problem. The cross validation error slowly decreases.

Uniform:

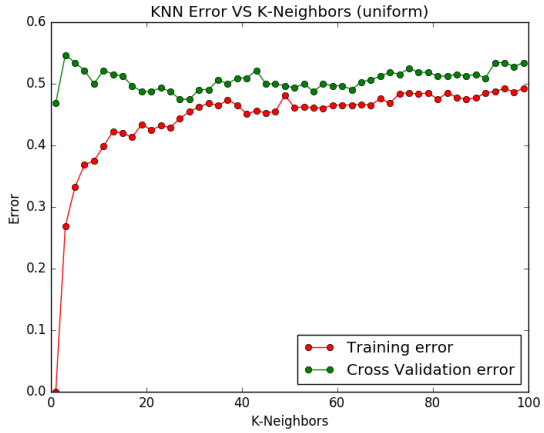


Figure 21. KNN Error VS K-Neighbors (Uniform)

Figure 21 shows the relationship between error and the number of neighbors. By comparing Figure 19 and Figure 21, we can find that the uniform weighting make training error more sensitive to the number of neighbors. This makes sense, because uniform weighting gives much more weight on the point that farther away than distance weighting method. When K is smaller than 20, there is overfitting, which means the testing error increases while training error decreases.

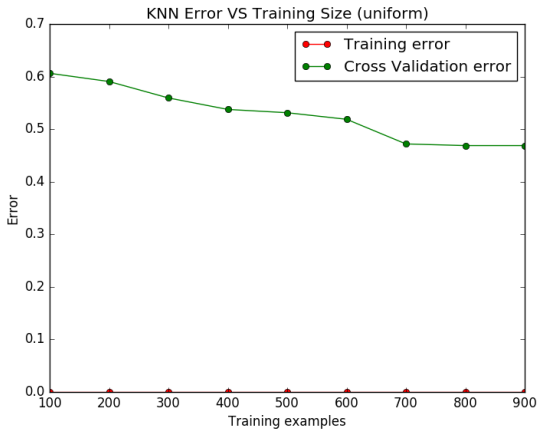


Figure 22. KNN Error VS Training Size (Uniform)\

Figure 22 shows the relationship between error and training sample size. The cross validation error slowly decreases. The training error was 0 all the time. I also tried KNN using other regression datasets, the training error was rarely 0.

2.4.2 Spam

Distance:

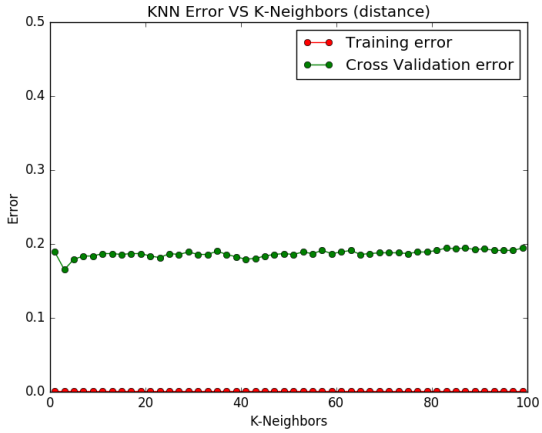


Figure 23. KNN Error VS K-Neighbors (Distance)

Figure 23 shows the relationship between error and the number of neighbors. For the training dataset, the error are always zero. For cross validation error, it almost shows no influence with the number of K, which is different from wine dataset. It may because that the data points are far way that has little weight to influence the error.

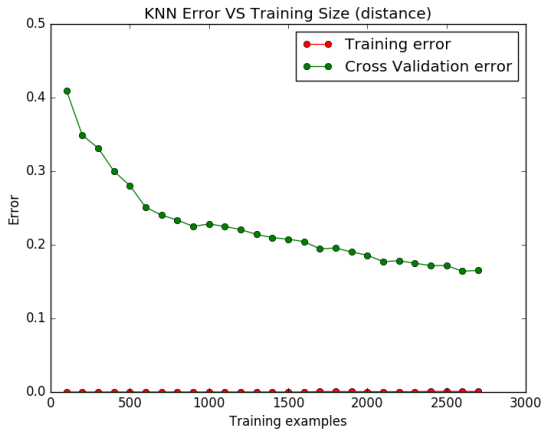


Figure 24. KNN Error VS Training Size (Distance)

Figure 24 shows the relationship between error and training sample size. The training error was 0. The cross validation error slowly decreases.

Uniform:

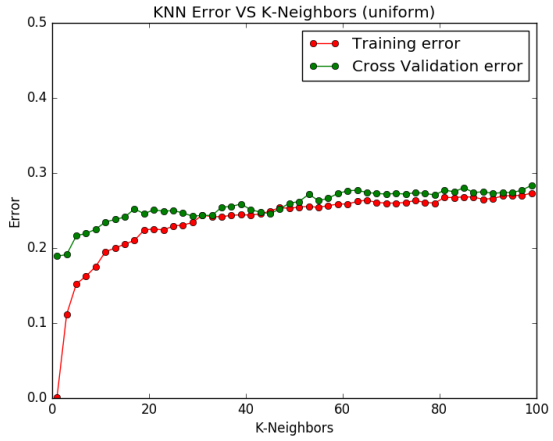


Figure 25. KNN Error VS K-Neighbors (Uniform)

Figure 25 shows the relationship between error and the number of neighbors. There is no obvious overfitting for this dataset.

Figure 26 shows the relationship between error and training sample size. The cross validation error slowly decreases. The training error was 0 all the time.

The spam problem is a binary problem, thus it's not hard for KNN to get 0 training error.

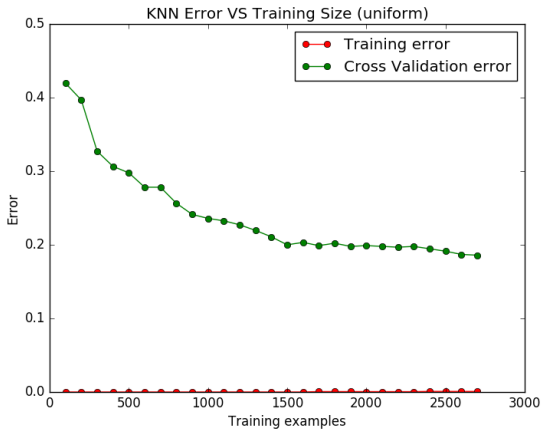
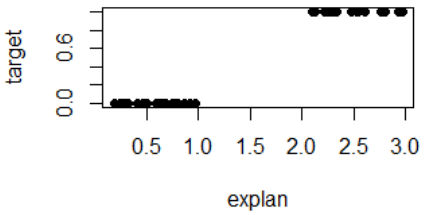


Figure 26. KNN Error VS Training Size (Uniform)

Based on he analysis of these 2 datasets, k-NN with k=1 does not always imply overfitting. It depends on the dataset. The picture below can easily explain the reason.



2.5 SVM

Two different kernels (linear and radial) were used to implement SVM. The performance of different kernels was evaluated. The kernel trick avoids the explicit mapping that is needed to get linear learning algorithms to learn a nonlinear function or decision boundary. When using linear kernel, the runtime is the shortest. Radial kernel needs longer runtime. SVM is eager algorithm, it takes much more long time to train if the kernel is complex.

2.5.1 Wine

Besides linear and radial kernels, I tried to use polynomial kernel for wine dataset, because it's smaller than the spambase dataset. However, the experiment took more than 4 hours. The runtime: linear < radial <

polynomial

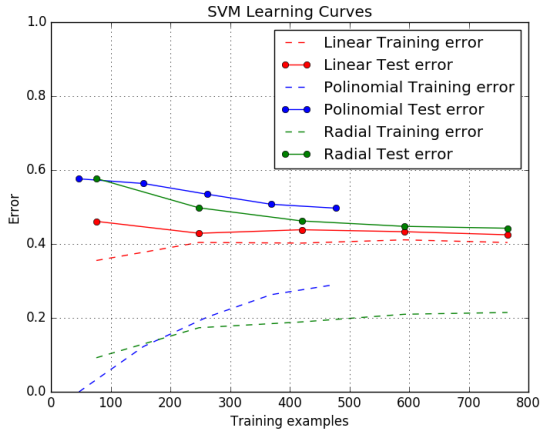


Figure 27. SVM Error VS Training Size

Figure 27 shows the relationship between error and training size. The test error drops with increased training set size. Usually, the training error should also decrease with the increase of training size, but in this case, the error slowly increased. Radial kernel provides better result.

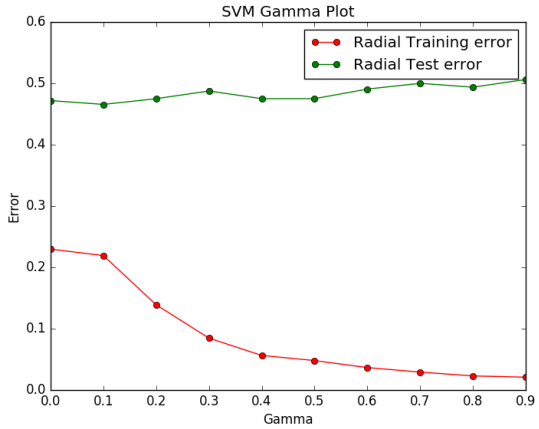


Figure 28. SVM Error VS Gamma

Figure 28 shows the relationship between error and gamma for the redial kernel. Gamma defines how much influence a single training example has. The larger gamma is, the closer other examples must be to be affected. As the gamma increases, the training error drops quickly, while the change of the test error is not that obvious.

2.5.2 Spam

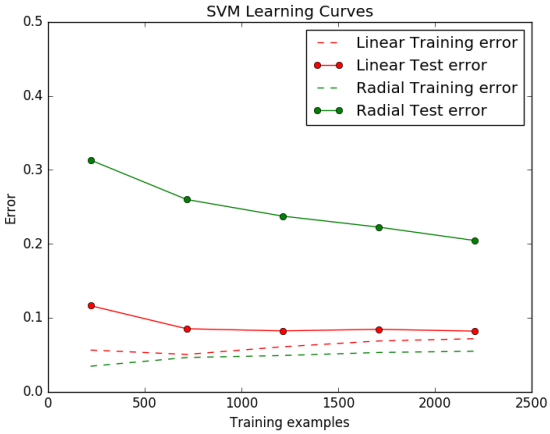


Figure 29. SVM Error VS Training Size

Figure 29 shows the relationship between error (calculate by error =1-accuracy) and training size. The test error drops with increased training set size. Similar to wine dataset, the training error increased with the increase of training size, while the test error drops.

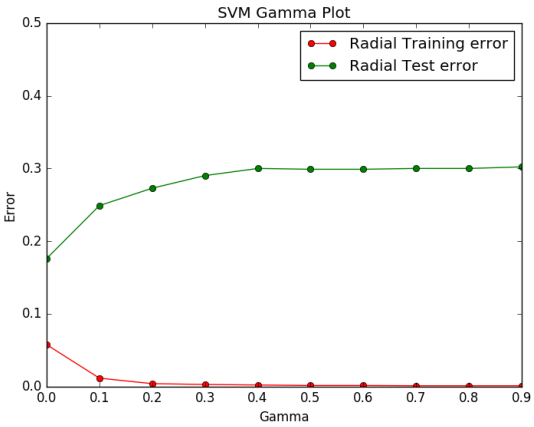


Figure 30. SVM Error VS Gamma

Figure 30 shows the relationship between error and gamma for the redial kernel. As the gamma increases, the training error drops quickly, while the test error increase.

3. Comparison and Conclusion

Figure 31 and Figure 32 compare the error (calculated by error=1-accuracy) across the different algorithms over the same training sets for wine and spam dataset. For wine dataset, though all the 5 algorithms have similar accuracy. KNN and boosting give the best

performance with short runtime. Thus, it's the most efficient analysis method for dataset 1. For the runtime, ANN ~ 300 s, SVM ~ 80 seconds, decision tree ~ 6 s, boosting ~ 70 seconds and KNN ~ 6 s to analyze the training set. The wine dataset cannot be analyzed very well by all of the different algorithms compared with spambase.

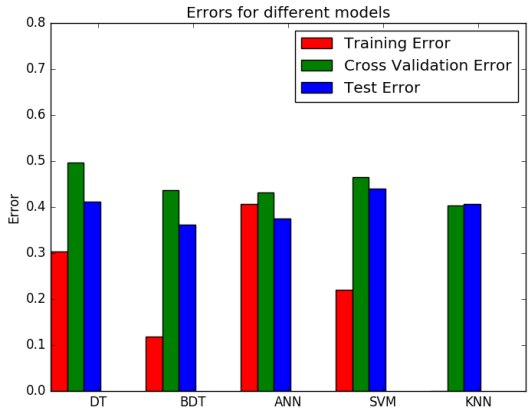


Figure 31. Wine: Errors of Different Models

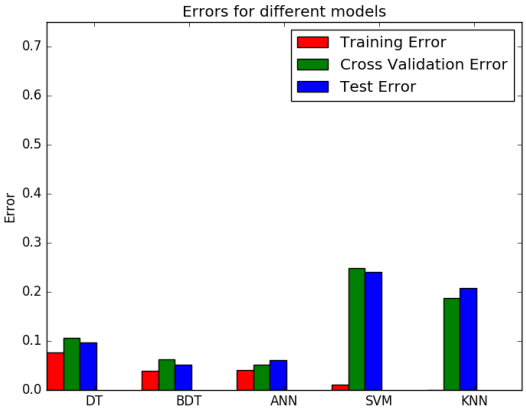


Figure 32. Spam: Errors of Different Models

For the spambase dataset, ANN, boosting have similar levels of error. For the runtime, ANN ~ 1200 s, SVM ~ 700 seconds, Decision Tree ~ 10 s, Boosting ~ 100 seconds and KNN ~ 15 s to analyze the training set. The best algorithm for this dataset with large number of attributes and missing values is boosting.

The table 1 shows the comparison between the algorithms [3].

Table 1. Comparison of KNN, Decision tree, AdaBoost and ANN

Algorithm	KNN	Decision trees	AdaBoost	Neural networks
Problem Type	Either	Either	Either	Either
Results interpretable by you?	Yes	Somewhat	A little	No
Easy to explain algorithm to others?	Yes	Somewhat	No	No
Average predictive accuracy	Lower	Lower	Higher	Higher
Training speed	Fast	Fast	Slow	Slow
Prediction speed	Depends on n	Fast	Fast	Fast
Amount of parameter tuning needed (excluding feature selection)	Minimal	Some	Some	Lots
Performs well with small number of observations?	No	No	No	No
Handles lots of irrelevant features well (separates signal from noise)?	No	No	Yes	Yes
Automatically learns feature interactions?	No	Yes	Yes	Yes
Gives calibrated probabilities of class membership?	Yes	Possibly	Possibly	Possibly
Parametric?	No	No	No	No
Features might need scaling?	Yes	No	No	Yes
Algorithm	KNN	Decision trees	AdaBoost	Neural networks

Reference

[1] Forrest D L K. Shrunken learning rates do not improve AdaBoost on benchmark datasets[D]. , 2001.
[2] Sun Y, Todorovic S, Li J. Reducing the overfitting of AdaBoost by controlling its data distribution skewness[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2006, 20(07): 1093-1116.
[3] http://www.dataschool.io/comparing-supervised-learning-algorithms/