CS 7461 Assignment 1

Supervised Learning

Yan-lin Chen, ychen3017, ychen3017@gatech.edu

1 Introduction

This assignment is to use 5 different supervised learning algorithms to run on 2 datasets. All of following experiments are run on i7-5600U laptop with algorithms from scikit-learn. This report will follow the sequence of dataset, model analysis, and summary.

2 Dataset

2.1 Wisconsin Diagnostic Breast Cancer (WDBC)

The reason to choose breast cancer data set is that my mother passed away due to breast cancer in 2002. I didn't have a chance to actually say goodbye to her. Now I am a father of a 3 year boy, sometimes I felt that I really missed my mother, but no chance to talk her now. Take this chance; I would like to know more about breast cancer, it is sort of a way that I could communicate with my mother. In addition, it is really appreciated for people that are doing research on how to classify breast cancer while it is still in early stage. Thousands of life could be saved. The data set has total 569 instances; it is binary classification problem, in which 357 instances are benign, 212 instances are malignant. So the baseline accuracy is about 62.7%. Test set is 30 % of dataset. Rest 70 % is the training set. I'.ve done stratified slit on the data to make sure the representativeness of test set and training set. There are 30 features for each instance. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass

2.2 Optical Recognition of Handwritten Digits (ORHD)

The reason to choose Optical Recognition of Handwritten Digits dataset is that I am quite interested in how machine can be trained to recognize images. I watched my son learning things little by little; sometimes I feel really amazing how my son recognize so many things in such a short time. Watching him learning more things, is really full of joy. So, I wonder what it is like for machines to learn to recognize different images as well. The dataset contain images of hand-drawn digits, from 0~ 9. Each instance has 64 attributes, with range from 0~16. There are 3,823 training samples, and 1,797 testing samples. It is quite well distributed from 0~9 for both training set and test set. Each number accounts about 1/10 of the dataset.

3 Model Analysis

3.1 K-Nearest Neighbors

For KKN algorithm, I used sklearn's KNeighborsClassifier to run my datasets.

3.1.1 Model Complexity Experiments

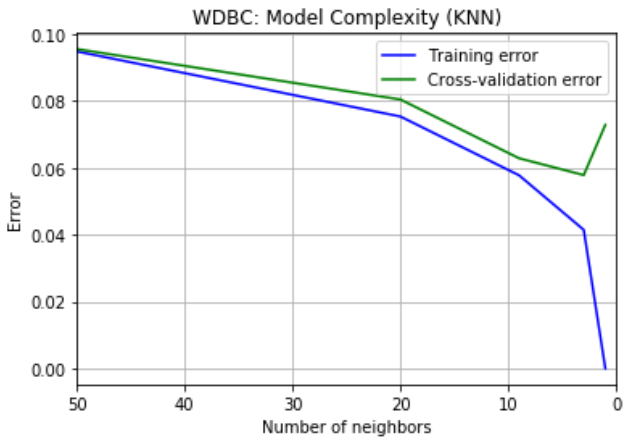

Fig 1

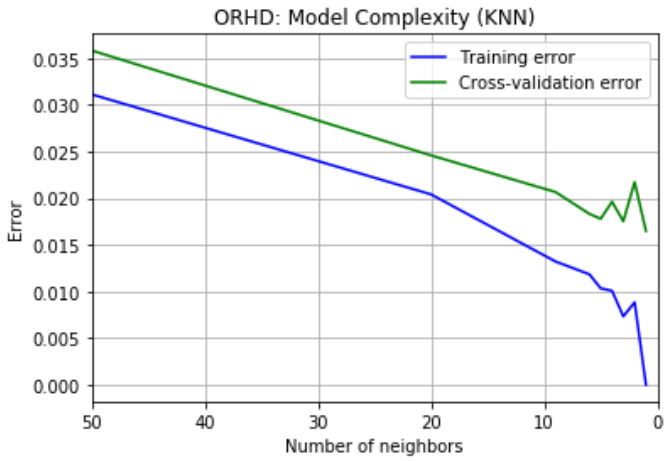

Fig 2

In the Model Complexity Experiments (WDBC), I use number of neighbors as the parameter of model complexity. More neighbors means the model is smoother or more simple, so the plot uses a reverted x axis. The cross-validation is in 5 folds, this also applies to the rest part of this report. Regarding to the suitable K value, I've tested with k== 1, 3, 9, 20, 50. From the result, the smaller the k is, the more it would become overfitting. It seems like it is better to have k==3. As to other parameters for KNN model, such as the weight distance, if change from default Uniform weights (each point has same weights) to Distance weights (close points have higher weights), the training accuracy result would increase from 0.957 to 1. This is a great improvement. It also means that if we find the attributes of a potential breast cancer patient has similar to what we have classified in database, it is very likely we can classy the new instance.

As to ORHD's complexity experiment, I used k== 1, 2, 3, 4, 5, 6, 9, 20, 50 for experiment. From the graph, we are not able to see the point where may have overfitting. I suppose it is because the labels in the dataset are quite separated pretty well. In this case, I will use k==1 for my knn algorithm to have the lowest cross-validation error. As to other parameters for KNN model, we've got the same training result whether it is Uniform weights or Distance weights. This is aligned with k==1 which has lowest error rate. So it doesn't matter on the weights parameters. I will keep default uniform weights for my model.

3.1.2 Learning Curve Experiment

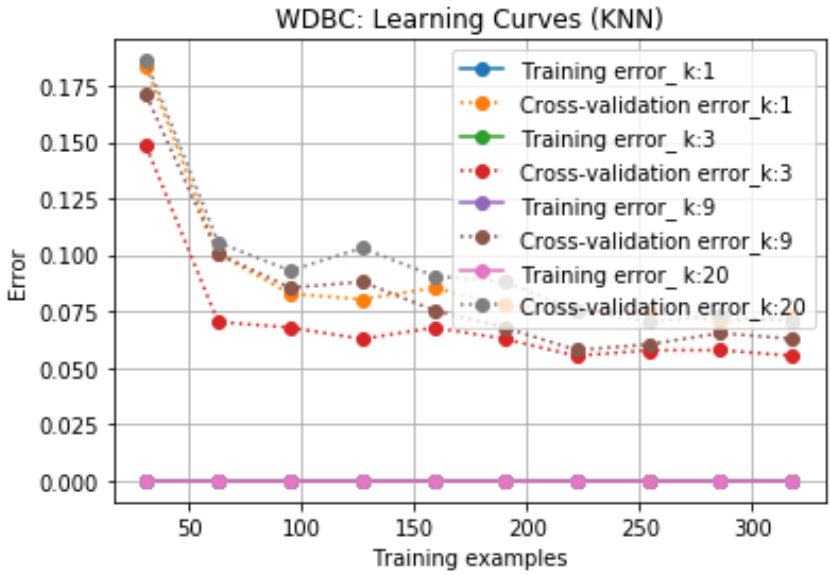

Fig 3

For WDBC's learning curve, it seems like that the error rate for cross-validation becomes stable at 0.06 when the training example is close to 125 from the learning curves above. It is good that we don't need that much data to have a good model for predicting breast cancer. In addition, I've also tested with different k again to see which k should perform best. Seems like k==3 is the right choice for our model.

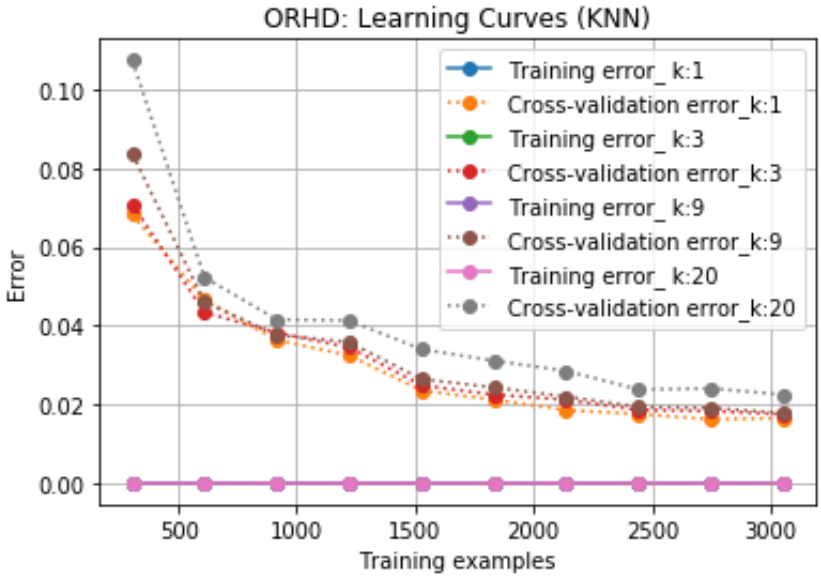

Fig 4

As to ORHD's learning curve experiment, we can see that k==1 has the lowest cross-validation error rate. The curve becomes stable when the training size is over 2,000 instances. If compare to teaching my son, I think maybe taught him less than 100 times before he can recognize 0~9. But relatively, it took much longer time (maybe few months) to teach my son. Guess those time spent by a child is also some kind of learning process. Or maybe he saw the numbers more than thousands of times through his eyes already.

Maybe I am only teach him how to pronounce the numbers, he already learned those numbers just don't know how to say them. Or maybe human brain's computing power is much larger than a computer. (Simulating 1 second of human brain activity takes 82,944 processors.)

3.2 Decision Tree

For Decision Tree algorithm, I used DecisionTreeClassifier from sklearn to run my datasets.

3.2.1 Model Complexity Experiments

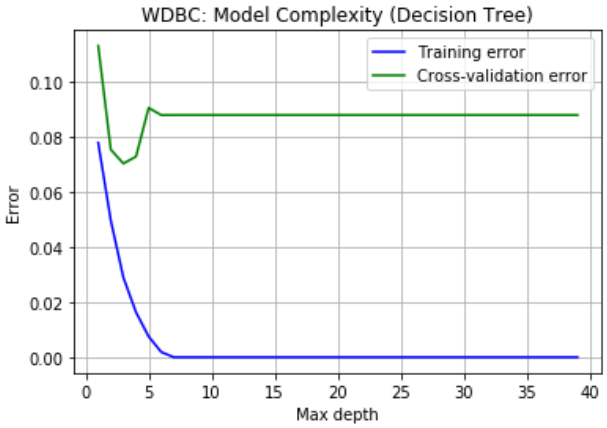

Fig 5

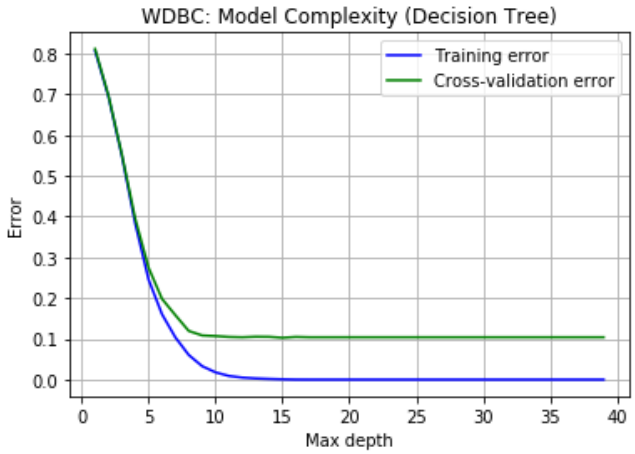

Fig 6

For WDBC dataset, max depth is used to measure model complexity. I've tested depth from 1 ~39 as above. The cross-validation error is lowest, around 0.07, when max depth is 3. The error rate goes up if the max depth increases. It is clear to see there will be overfitting situation. So for the following analysis, I've set to pre-pruning the tree with max depth of 3 to prevent overfitting. In addition, I've test with GINI impurity and information gain as different ways to measure the quality a split. Information gain has better result (about 1%) on training set. Therefore, I will use information gain for my decision tree model.

For ORHD dataset, I've also tested depth from 1 ~39 as above. It is clear to see that the error rate for cross-validation curve becomes stable when the depth reaches to 10. Consequently, the max depth for ORHD decision tree will set as 10. As to the measure of information after split, using information gain as measurement has better training result against using GINI impurity. (About 1.7% better)

3.2.2 Learning Curve Experiment

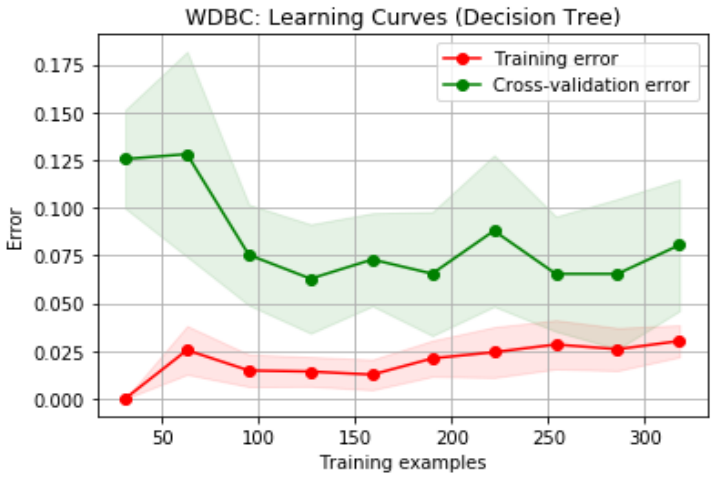

Fig 7

From the WDBC learning curve experiment result, learning curve for cross validation becomes lowest when training size is about 125. It is similar as if we use previously KNN model. We can see that both curves tend to merge together. In addition, I also checked what the important attributes for my decision tree model are, surprisingly it turns out that only 6 of the total 30 attributes have impact on the decision tree model. Following shows the decision tree graph. And among them, worst_perimeter is the most important one.

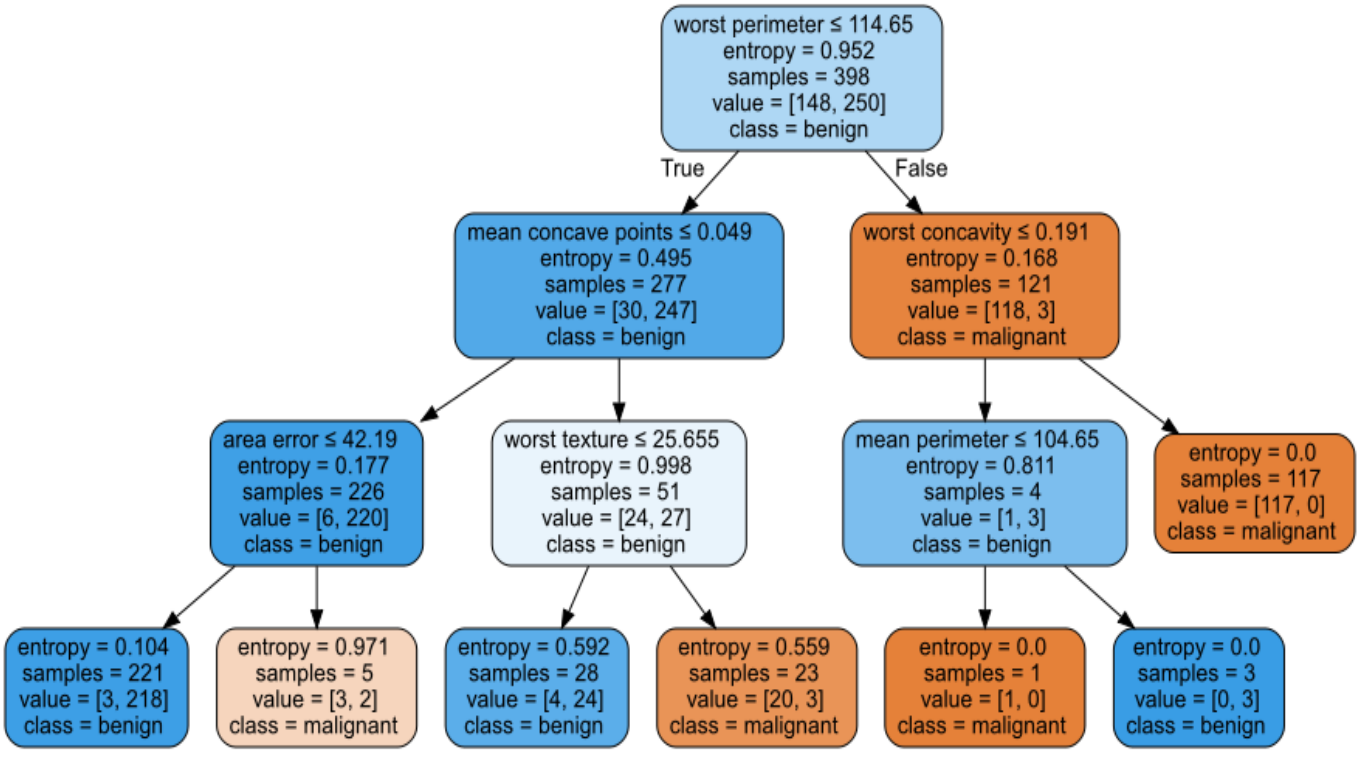

Fig 8

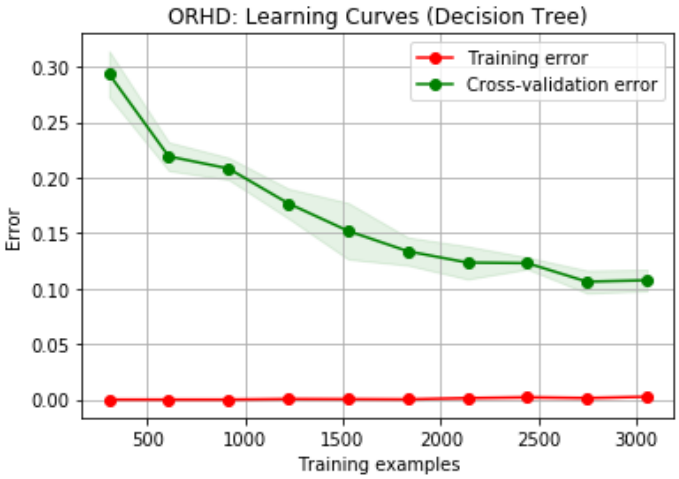

Fig 9

For ORHD dataset, the learning curve of cross-validation keeps going down, but the learning curve for training remains very close to 0. It looks like it is possible to lower the cross-validation error if we have more training examples.

3.3 Boosting

For boosting, I used AdaBoostClassifier from sklearn to build up the model. Decision tree is used as base estimator for AdaBoostClassifier.

3.3.1 Model Complexity Experiments

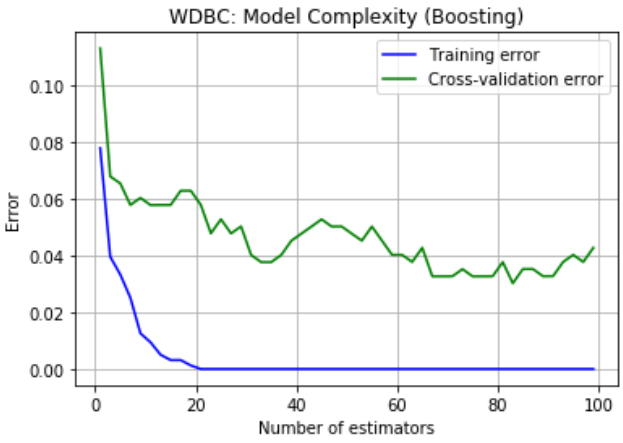

Fig 10

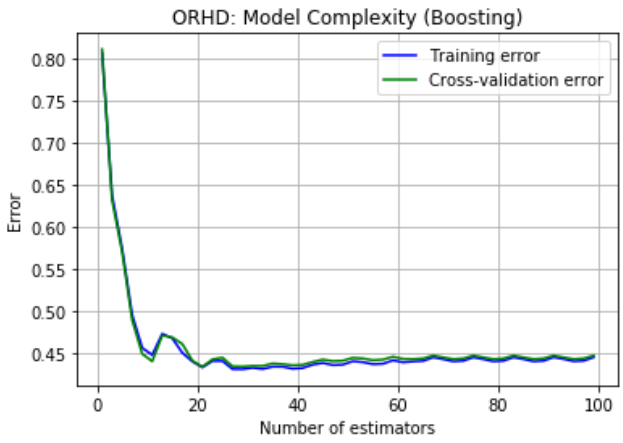

Fig 11

For WDBC Model Complexity Experiments, I use different number of estimators to measure model complexity. Before 60 estimators, the error rate drops quickly. After that it just decreased slowly to lower than 0.04. Therefore, I would set the number of estimators at 60 for Adaboost model as pre-pruning. In addition, we will not be able to see overfitting in AdaBoost model because of the combination of weak learners could avoid such a situation but we might have over-high confidence instead.

As to ORHD dataset, it seems like it is better to set the number of estimator at 20 for pre-pruning AdaBoost model. Yet, the error rate is very high compare to previous KNN model and Decision Tree model. Therefore, I've tried to twist different hyper parameter to improve the performance. I found that if I change the algorithm of weighing weak learner from SAMME.R to SAMME, the accuracy would increase by 7.2%. So seems like it is better for the weak learner to be weighed by discrete method. Furthermore, I went on to test different learning rate. Yet, there is trade-off between learning rate and number of estimators, I've tried several different combinations and listed the result as follow. It is better to set number of estimator at 50 with learning rate at 0.6. Learning rate small than 1 could also mean that the weak learners of ORHD data set is relative weaker than the weak learners of WDBC dataset.

Number of estimators (default 50)	Learning rate (default 1.0)	Accuracy (training)
20	0.5	0.457090528519
20	0.8	0.526425954997
20	1.1	0.548403976975
50	0.8	0.684720041863
50	0.7	0.779173207745
50	0.6	0.781789638932
50	0.5	0.725013082156
60	0.6	0.800104657248
60	0.5	0.65096807954

Table 1

3.3.2 Learning Curve Experiment

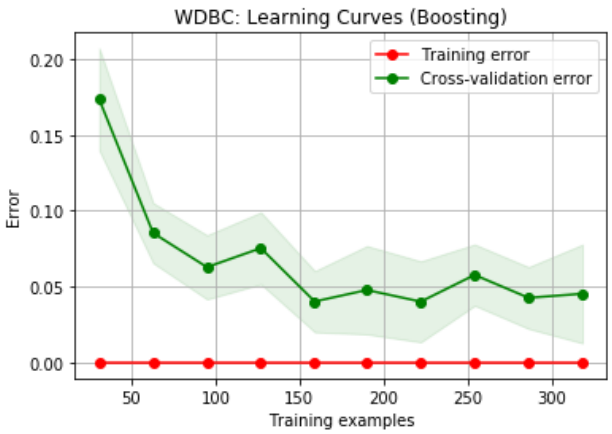

Fig 12

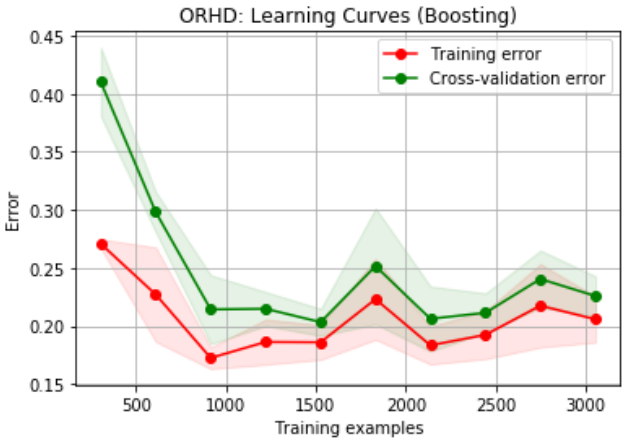

Fig 13

For WDBC dataset, the training error curve remains 0 as the training examples increased. For the cross-validation error curve, it becomes stable at 0.05 when training examples reach to 150. It requires a bit more training sample than KNN and Decision Tree.

As to ORHD dataset, both curves seem to merge together after training examples reach to 1,500. Compare to previous KNN & Decision Tree model, AdaBoost would require more effort to find suitable hyper parameters (such as learning rate, number of estimators, and algorithm) for the model.

3.4 Neural Networks

For Neural Networks, I use sklearn's [MLPClassifier] to run my datasets.

3.4.1 Model Complexity Experiments

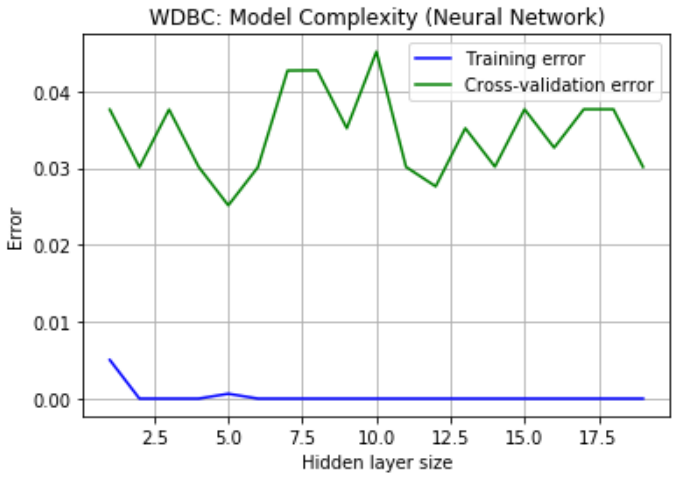

Fig 14

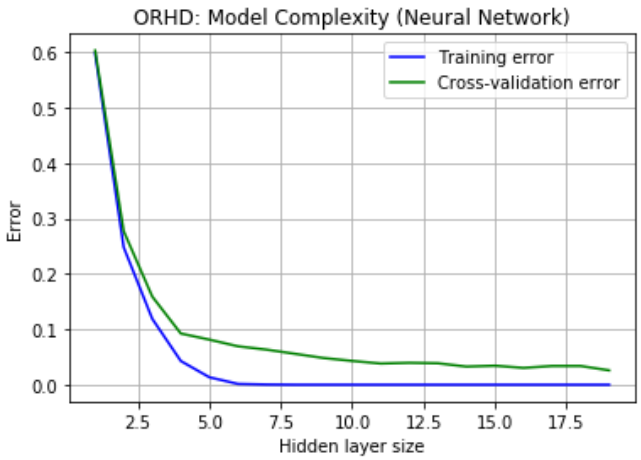

Fig 15

In this Model Complexity Experiments, I use hidden layer size as control parameter and all the Neural Network models here only contained 1 hidden layer. The difference is just how many neurons in the hidden layer. For WBDC dataset, the solver I used is "L-BFGS", which is suitable for smaller dataset. It seems like the error rate is lowest when then neuron in the hidden layer is 5. The reason not able to see overfitting is that MLPCalssifier has an alpha term to avoid overfitting.

For ORHD dataset, I use same setting as for WBDC dataset. From the figure, both curves become stable when hidden layer size is about 10. The error rate is about 0.5%. As ORHD dataset is about 9 times larger than WDBC, I tried with different two solvers, SGD (stochastic gradient descent) and adam (another stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba). However, both solvers are not able to converge within max 200 iterations. Thus, I keep using L-BFGS for my model.

3.4.2 Learning Curve Experiment

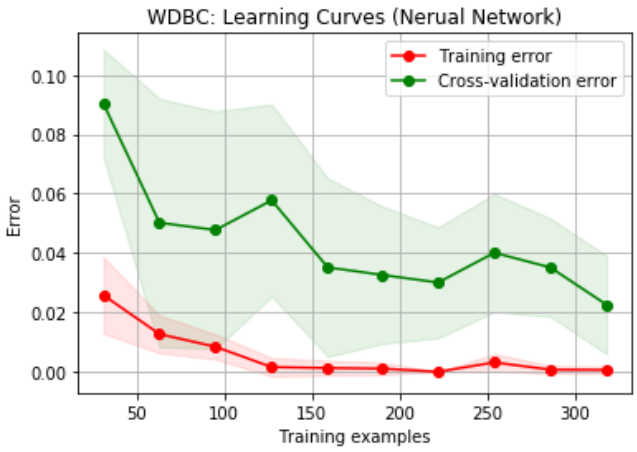

Fig 16

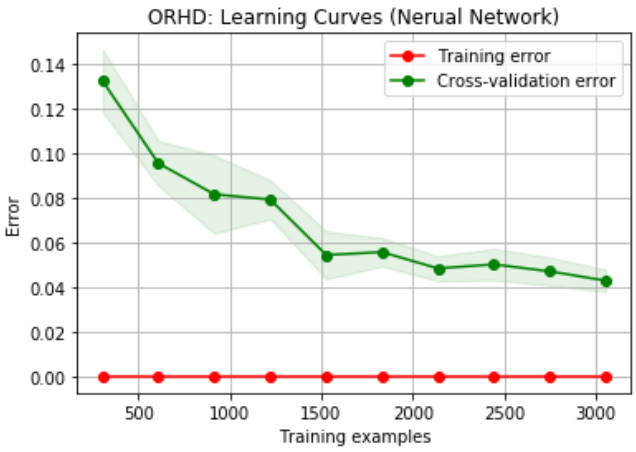

Fig 17

For WDBC dataset, the curve seems possible to go down further if we have more training examples. As mentioned above, we will not able to see overfitting in learning curves as well. In addition, it took total 84 iterations to reach convergence.

For ORHD dataset, we can see the two curves seem to merge, but still has gap. Meaning more training examples might help to increase model performance. Meanwhile, it took 52 iterations for the solver to find convergence.

3.5 Support Vector Machines

For SVM algorithm, I use SVC from sklearn.

3.5.1 Model Complexity Experiments

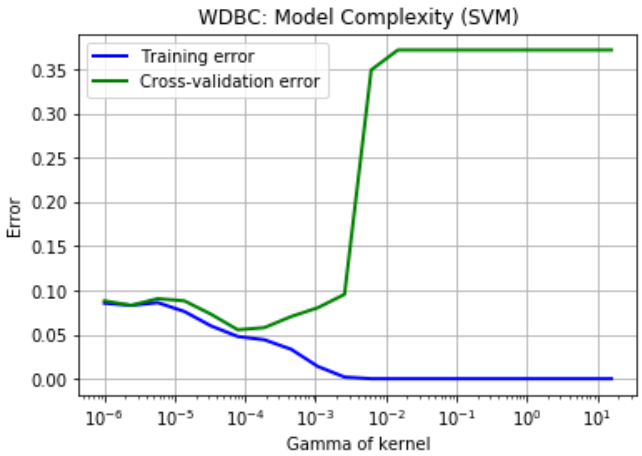

Fig 18

In Model Complexity Experiments for WDBC, firstly, I set the kernel as default RBF and tried with different gamma to make the curves. Gamma is the coefficient of radial basis function kernel in SVC. From

the graph, we can see that CV curve has lowest error when gamma=10^{-4} (about 0.0001). After this point, cross-validation error curve goes up clearly which would cause overfitting. Secondly, I tried with linear kernel. Accuracy and run time comparison as follow, accuracy quite close, but the run time is very different.

Kernel	Accuracy (training)	Run time (sec)
Rbf	0.952261306533	0.015006065368652344
Linear	0.969849246231	2.3036563396453857

Table 2

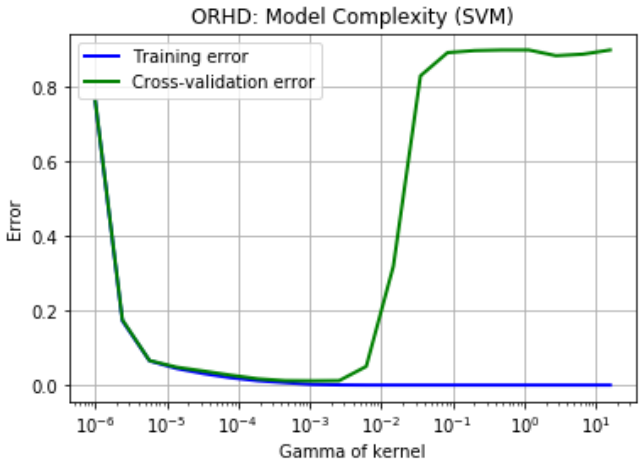

Fig 19

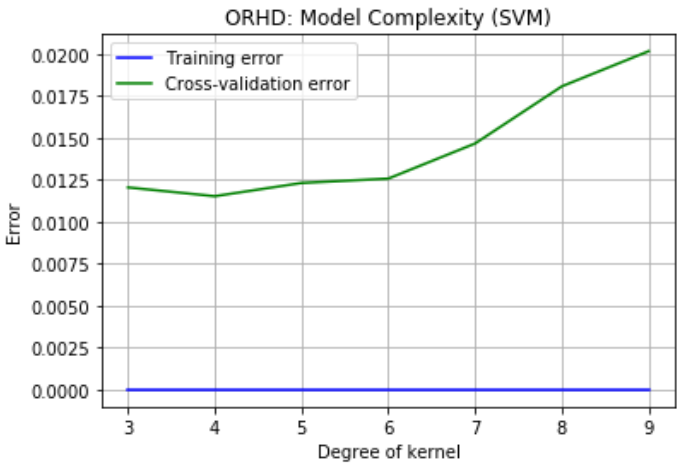

Fig 20

For ORHD dataset, firstly, I have same setting as for WDBC dataset as well. It is clear that CV curve has lowest error when gamma=$10^{-2.6}$ (about 0.0025). Same as WDBC, the CV error curve goes up clearly right after the tipping point. Secondly, I changed to kernel to poly, and tested with different degrees. From the figure, CV error curve has lowest error when degree is 4. Lastly, I use linear kernel for to fit the training dataset. Comparison table of these 3 kernels as follow. Accuracy is almost the same after choosing suitable gamma and degree for correspondent kernels. But the run time for Rbf is much longer compare to that of Poly and Linear kernel. (Note: I was trying poly kernel on WDBC as well, but it took too long to run the experiment.)

Kernel	Accuracy (training)	Run time (sec)
Rbf	0.999476713762	2.8420209884643555
Poly	1.0	0.5173659324645996
Linear	1.0	0.46933579444885254

Table 3

3.5.2 Learning Curve Experiment

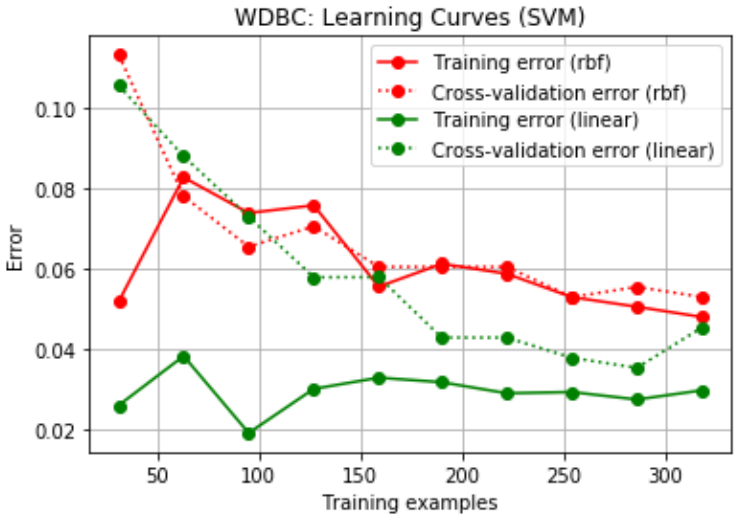

Fig 21

For the learning curve experiment of WDBC dataset, I have compared rbf and linear kernel on the same graph. We can see that both kernel's learning curves have tend to merge together. Rbf kernel merge quicker at when traiing examples reach to about 50. Linear kernel merged latter when traing examples reach to about 200. However, the error rate for linear kernel is much lower. Consdering we are doing breast caner diagnostic classification problem, which could impact people's life dramatically, though the running time of linear is longer, I would still prefer to use linear kernel for my model.

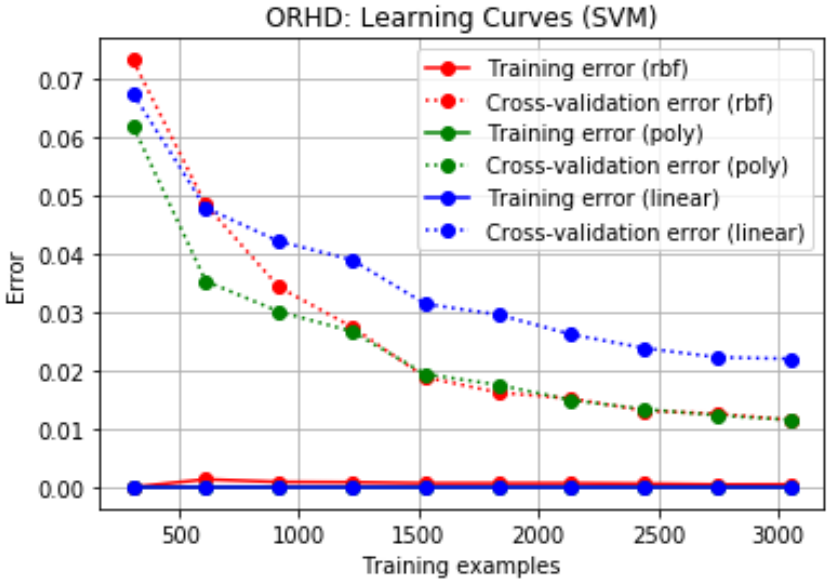

Fig 22

For ORHD dataset, further complexity experiemnts. I put three kernels' learning curves on the same graph together. All three cross-validation error curves have same downward trend. However, poly and rbf kernel have lower error rate than linear kernel. Combine the experiments result from complexity experiments, I would go for poly kernel with degree at 4 for my model, as it has highest accuracy with relative lower run time.

4 Summary

Dataset	Algorithm	Accuracy (Training)	Accuracy (Test)	Training examples	Run Time (sec)
WDBC	KNN	1.0	0.918128654971	125	0.09706902503967285
WDBC	DT	0.969849246231	0.947368421053	125	0.015011787414550781
WDBC	Boost	1.0	0.988304093567	150	0.2832043170928955
WDBC	NN	0.997487437186	0.953216374269	300 up	0.05704045295715332
WDBC	SVM	0.969849246231	0.953216374269	200	2.903080701828003
ORHD	KNN	1.0	0.97995545657	2,000	1.175840139389038
ORHD	DT	0.997383568812	0.871937639198	3,000 up	0.17912769317626953
ORHD	Boost	0.800104657248	0.753897550111	1,500	0.774566173553466
ORHD	NN	1.0	0.936525612472	3,000 up	0.5583972930908203
ORHD	SVM	1.0	0.975501113586	3,000 up	0.5343782901763916

Table 4

For WDBC dataset, considering accuracy, testing examples, and run time, KNN, AdaBoost, and Neural Networks are relative better compare to Decision Tree and SVM. If I have to choose one, probably I will go for KNN, as it provides best accuracy and with less training examples.

For ORHD dataset, AdaBoost clearly works not as well as other models. Especially, this training result was after a lot of hyper parameters' adjustment. Therefore, it would actually take more time than just simply the running time. I think AdaBoost is not suitable for image like dataset.

From model comparison point of view, KNN is relative easy to understand, but it requires longer time to run. Decision Tree runs most quickly on both datasets and relative easy to implement, which might be suitable to start with for classification problem. Another good thing about Decision Tree is that you can actually see which attribute matters; it may help us to reduce the effort to collect the data in the future. (From 30 reduce to 6 in WDBC data set.) As to AdaBoost, it will be critical to find a proper balance between learning rate and number of estimators. As to Neural Networks, though the run time on both datasets is not long, but it is possible that the model never reach to convergence during my experiments. In addition, it is quite difficult to understand what the reasoning is behind, it may not be relevant to image like data, but it may not be good for data like WDBC. For SVM, kernel selection would be critical for finding the proper model.