SUPERVISED LEARNING (HW #1)

Machine Learning (CS 7641)

Dominic Frecentese

February 10, 2019

INTRODUCTION

This report seeks to assess the performance of a number of learning algorithms (Decision Tree, Boosting, K-Nearest Neighbors, Neural Network, Support Vector Machine) on two different classification problems to better understand the characteristics of the learners. We will begin by explaining the selected classification problems and why these problems are interesting for learner comparison. We will then explore the various learners and describe the methodologies and parameter selection for each, including performance analysis and commentary for the learners over the learning problems.

PART 1: CLASSIFICATION PROBLEMS

MAMMOGRAPHIC MASS

The first classification problem is to predict whether a mammographic mass is malignant or benign. The features available to the learners for predicting this outcome are: patient age, mass shape, mass margin and mass density. This is a binary classification problem; there are two possible outcomes.

The dataset has 960 examples, all of which are labeled. Some of the features have missing values. To ensure the missing values did not affect learning I chose to remove any example that had a feature with missing values. After this cleaning, the dataset was pruned to 831 examples.

This is an interesting classification problem for both practical and theoretical reasons. From a practical perspective, a large number of breast mammograms lead to unnecessary biopsies (70% of these breast biopsies are benign). If there were a way to more accurately predict whether a mammogram is malignant or benign, money could be saved on biopsies, and patient strain could be lessened.

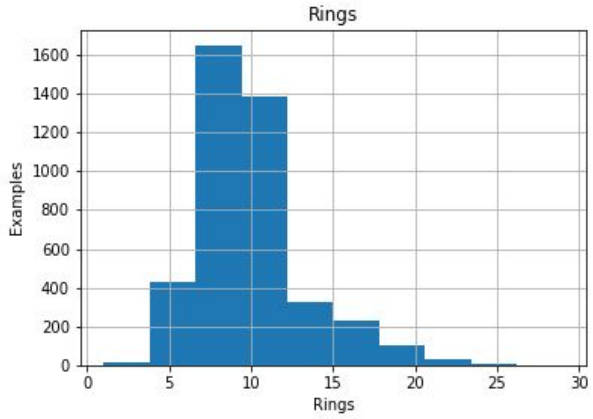
From a theoretical perspective, the number of examples available to the learner is quite small at less than one thousand. This will allow us to see if any learners perform particularly well or poorly with a small example set. Additionally, the number of features or inputs for each learner is also quite sparse at four. It is a binary classification problem, which was chosen as a complement to the multi-label classification of the Abalone problem (explained in more detail below).

Finally, this problem strikes a good balance between being predictable but not overly simple. The initial decision tree built for this problem had an accuracy of ~75% on the test set. A random approach would give 50% accuracy, and perfect predictive ability is 100%. This baseline accuracy provides a nice range for comparing different learners.

ABALONE

The second classification problem is to predict the age of Abalone fish based on various attributes of the fish: sex, length, diameter, height and a number of weight measurements. In total there are 8 features that are used for learning and one target feature. Given there are multiple values of abalone age, this is a multi-label classification problem. The target feature Rings is equal to age minus 1.5. Therefore to get Abalone age we add 1.5 to the Rings feature.

There are 4177 total examples in the dataset and no missing attribute values, thus all examples are retained and used on learning. The distribution of samples for the target feature (Rings) follows a roughly normal distribution with long tails on both the low and high end. This can be seen below:



There are 30 different values for the target label ‘Rings’. Given the sparse availability of examples in the high and low range of ‘Rings’ (there are <25 examples for each ring of ring size <5 and >15) I chose to discretize the target label by creating buckets for Rings. This ensures that there are enough examples for the learners to learn for each label value. The updated buckets for Rings are: 1-5, 6-10, 11-15, and 16+. The distribution after applying discretization:

Rings-B	Count
1-5	189
6-10	2541
10-15	1186
15+	261

This classification problem is less interesting from a practical perspective, but more-so from a personal perspective. I am fascinated by the natural world, the patterns that underpin much of it, and the ability to use data to guide biological understanding and environmental protection. If we are able to successfully predict abalone age based on physical characteristics, we showcase another underlying pattern and understand this creature a little bit better.

The Abalone problem and dataset is interesting form a theoretical perspective as it is difficult for the learners to have good performance. Even with the discretized target feature, model accuracy for the Decision Tree learner is ~71%. If we were predicting the Rings feature directly, learner performance would drop precipitously.

These two classification problems are especially interesting when viewed as complements. The mammogram dataset has relatively few examples (as discussed above) while Abalone has a around 5x as many. Additionally, the Mammogram set has 4 features for learning while Abalone has 8. Lastly, Mammogram is a binary classification problem while Abalone is multi-label classification problem. These three differences will give the learners more variety to showcase their differences, and are why I selected these two classification problems together.

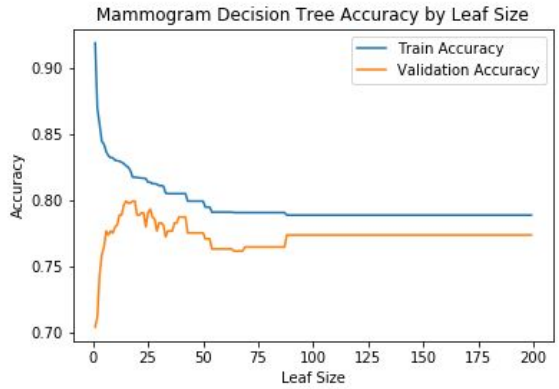
Feature scaling was applied to both datasets to ensure that features with naturally higher magnitudes were not overly weighted in algorithms like KNN and Neural Networks.

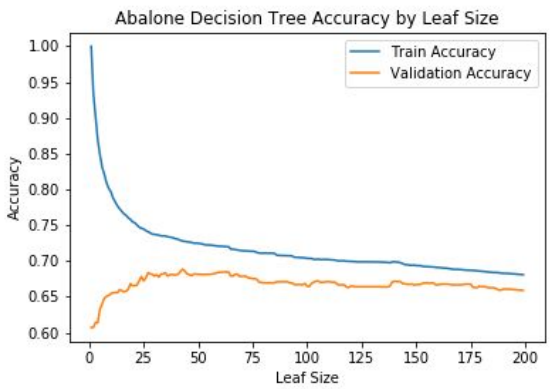
PART 2: TRAINING LEARNERS

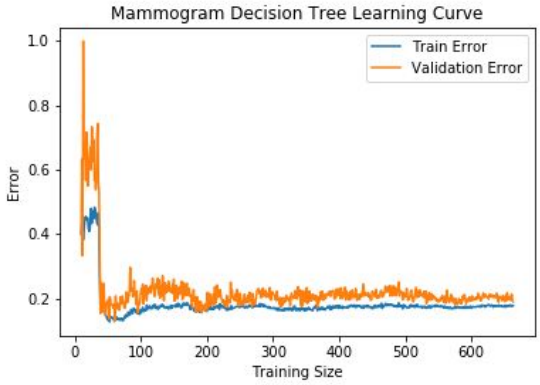
All of the following learners were trained using modules in the sklearn python package. Validation accuracy across all learners is model accuracy using a 5-fold cross validation approach. Each model will begin with a brief overview on methodology and parameter selection followed by analysis. Finally we will conclude with a comparative analysis across all learners.

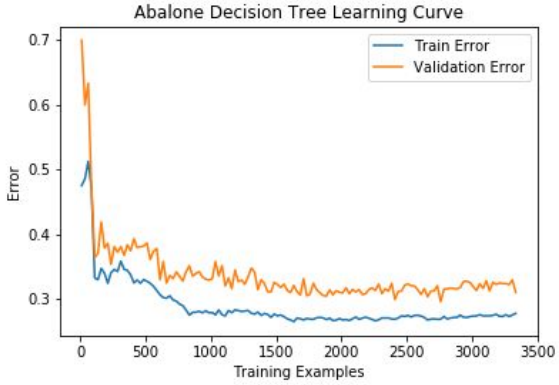
DECISION TREE

The hyperparameters that need to be tuned for the Decision Tree learner are leaf size, node depth and split criterion. I chose to use GINI index for the split criterion opposed to Entropy. The GINI index is a measure of the homogeneity of a classification split in each resulting group. The remaining two hyperparameters effectively serve an identical purpose (how granular do we build the tree), I choose to focus on optimizing leaf size. To determine the optimal leaf sizes I built DT models for leaf sizes of 1-200, and assessed performance of each model with a 10-fold cross validation approach. The results for each dataset can be seen below:









Mammogram Mass

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Decision Tree	81.9277	74.8503	0.00119901	0.000138998

Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Decision Tree	72.6729	70.933	0.015223	0.000536919

There are a number of interesting things to note about the decision tree learners and the above results. First, it's clear when overfitting occurs with respect to leaf size in both problems. For mammogram mass, this occurs at a leaf size of about 14. With leaf sizes any lower, it is clear that the training accuracy increases and validation accuracy decreases. With our Abalone problem, the trend is similar but the optimal leaf size is much higher at 43. The difference between these two leaf sizes leads me to believe that there is more noise in the Abalone problem, as less "specificity" is used to create the optimal model.

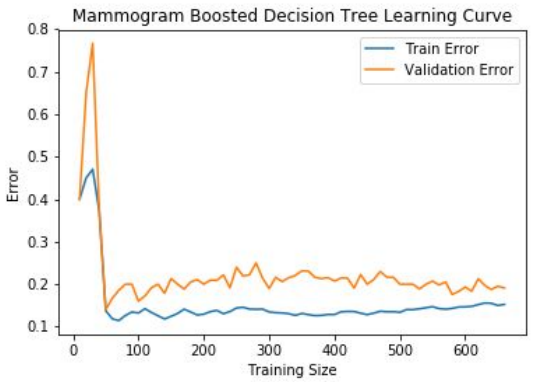
It was also interesting to see that varying the values of leaf size didn't have a massive effect on the performance of each model. There is about a 2.5 percentage point increase in validation accuracy for the mammogram problem between the best and worst leaf sizes (excluding overfitting territory). The delta for Abalone is very similar at 2-3 percentage points. There is a large improvement in model performance (10 percentage points) when factoring in the overfitting range.

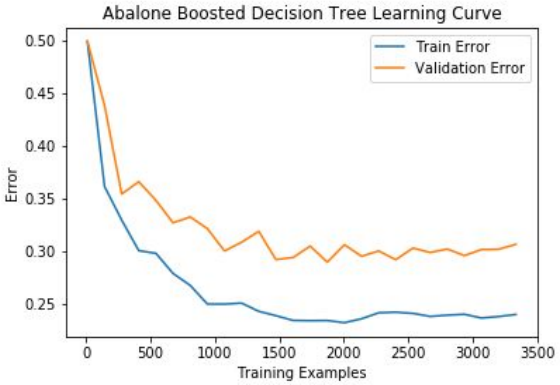
For the mammogram problem, it's interesting that after 50 examples, the learning curve is relatively flat. In other words, it only took 50 examples for the DT learner to create an effective model, and examples beyond this point do not improve the model. On the other hand, the DT model for Abalone is improving through ~2k examples, then remains flat from 2k-4k examples.

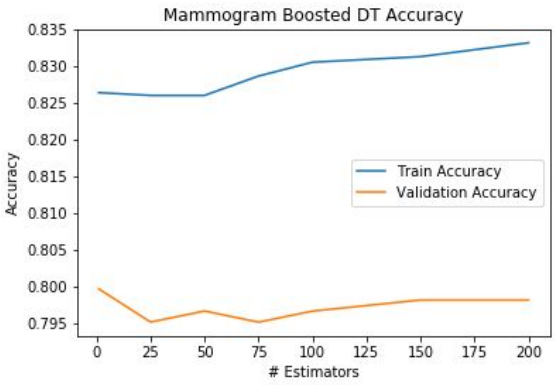
For improvements to the Decision Tree learner, I would want to explore using the GINI index vs entropy as the splitting methodology. I chose GINI index after reading resources online and this seemed to be the standard, but it would be helpful to put that to the test.

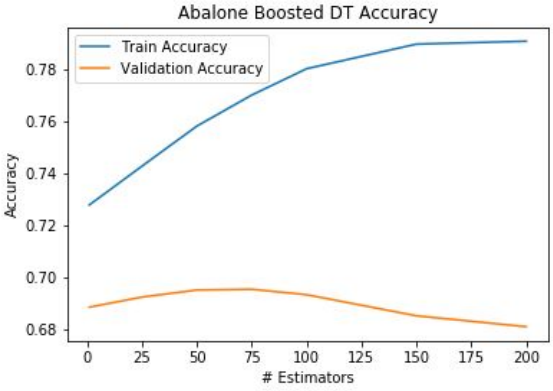
BOOSTING

For boosting on top of my Decision Tree I used the Adaboost algorithm, which iteratively trains the base decision tree learner by putting more weight on the mis-classified examples in each iteration. The hyperparameters considered and tuned in Adaboost are the learning rate, the number of estimators and the decision tree depth. Since I built Adaboost learner on top of the already pruned decision tree learner created above, determining the DT depth has already been taken care of. To determine the optimal combination of learning rate and number of estimators, I used the GridSearchCV module in sklearn.









Mammogram Mass

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Boosting (DT)	83.2831	76.6467	0.433827	0.0303068

Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Boosting (DT)	75.5462	71.0526	0.870696	0.042778

The optimal hyperparameters for the Mammogram problem were num_estimators=200 and learning_rate=0.001. The optimal hyperparamters for the Abalone problem were num_estimators=50 and learning_rate=0.01.

The boosting results are most noteworthy when looked at in conjunction with the decision tree results, specifically in terms of training times and model accuracy. The boosting learner took substantially longer to train than a single decision tree. Intuitively this makes sense as the boosting algorithm is creating and training a decision tree at each iteration. What's most interesting though is that the training time of the boosting learner is greater than simply multiplying the number_estimators times the decision tree training time. For example, in our Mammogram problem, the single DT train time = 0.001 secs. Multiplying by the number of boosting estimators (130) gies 0.130 secs. However, the boosting train time was almost 4x as higher than this at 0.43 secs. I believe that this is caused by additional algorithmic steps in Boosting that aren't included in Decision Tree learning such as the calculation of example weights after each boosting iteration.

Boosting performed slightly better than the Decision Tree learner in both problems, however the increase in accuracy was not as great as I expected. The test set accuracy of the boosting algorithm was about 2 percentage points higher than decision tree on the Mammogram problem and 0.1 percentage points better on the Abalone problem. The learning curves by iteration for boosting agree with this result, from the graphs it can be seen that validation accuracy is only slightly improving as we increase the number of estimators (iterations) in the boosting algorithm from 1->200.

There are two improvements I would consider to improve performance of the boosting algorithm. First is to try using stumps, or unpruned decision trees, rather than the pre-pruned decision tree optimized from our first learner. The current methodology is combatting overfitting at 2 steps, first by optimally pruning the base DT, and then by applying Boosting. We may be finding a non-globally optimal solution by finding optimal solutions at two different steps in the process, rather than allowing Boosting to try different tree depths in it's base learner. The second improvement I would explore is using other boosting algorithms such as GradientBoost.

K-NEAREST NEIGHBORS

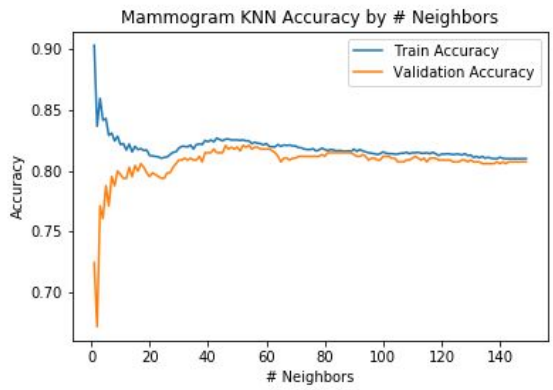
The hyper-parameters that need to be considered with a K-Nearest Neighbor algorithm are: the number of neighbors (K), the distance metric and the weighting strategy for the K-Nearest points. I decided to use euclidean distance metric and a uniform weighting for the K-nearest points and focus on the optimal value of K. I trained 150 KNN learners with values of K from 1 to 150, then calculated accuracy for each learner using 5-fold cross-validation. I then used the value of k from the learner with the best cross validation accuracy (including visual confirmation that no overfitting was occurring). The results of model performance by varying K are shown below for both classification problems.

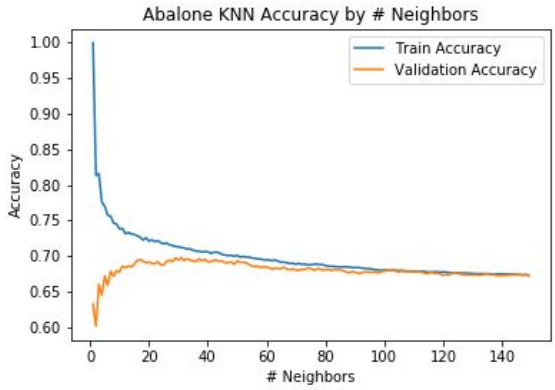
Mammogram Mass

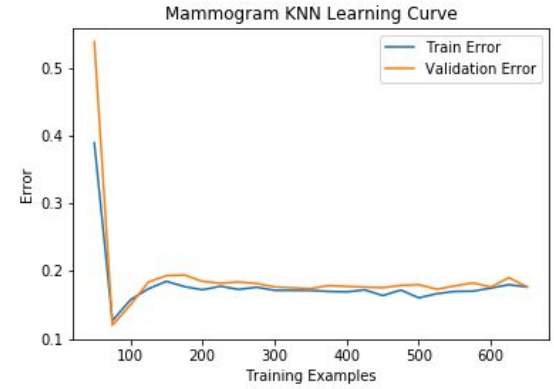
Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
KNN	82.6807	77.8443	0.00157309	0.0188339

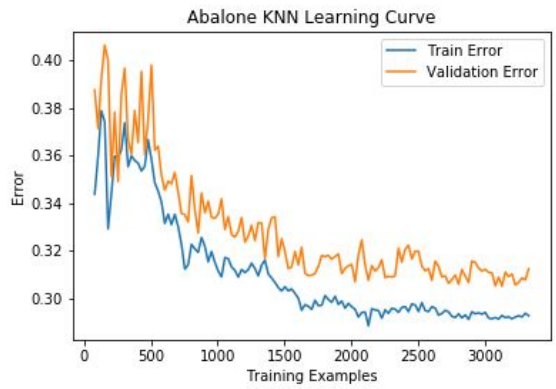
Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
KNN	71.6851	70.0957	0.00476599	0.132865









With this approach, the optimal value of K was found to be 52 for the Mammogram Mass dataset and 31 for the Abalone Age dataset. It's unsurprising that K is larger for the Mammogram mass problem as there are only two possible values of the target feature. All else constant, the more values there are for the target feature, the smaller I expect K to be as the total # of "neighborhoods" increases. This is confirmed in the above results as well.

The second attribute to note from the KNN results is the stark difference in query time vs train time. With the learners so far (decision tree, boosting) train time has dwarfed query time. This makes intuitive sense as both of the previous learners were eager learners, or learners that create target functions as soon as data is available to them. KNN on other hand is a lazy learner, and waits until query time to form a hypothesis representation. This can be seen in the train/query times for both problems. Query time is about 10x that of train time for the Mammogram problem, and 25x that of train time for the Abalone Age problem.

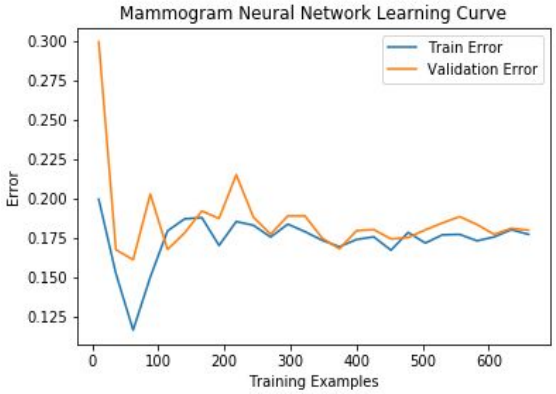
Examining the learning curves with respect to K, or the number of neighbors, it's also interesting to note that once the learners get out of overfitting territory (e.g. #neighbors < optimal #), the

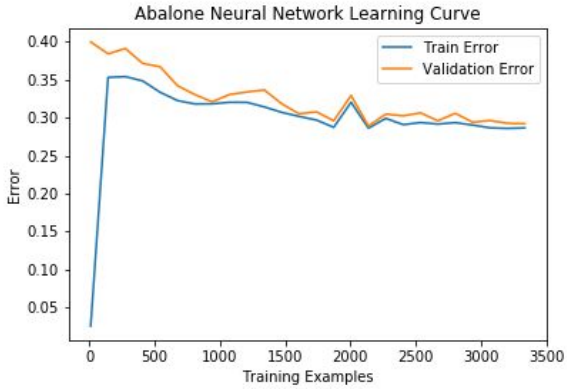
value of K doesn't have a large effect on the ultimate model accuracy. In the Mammogram problem, moving from k=50 to k=150 only reduces validation accuracy by ~1.5 percentage points. The story is similar for the Abalone problem and dataset. I did not confirm this, but I believe that if we extended the KNN learning curve with respect to K, we would eventually see a drop off in accuracy. The reason I suspect this is that eventually we would begin to expand the neighborhood into areas that have higher concentrations of other label values. In both problems, if we increased k to the size of the dataset, the learner would take on the value of the target feature that occurs the most often (assuming we continue to use uniform waiting). If I were to continue exploring KNN applied to these problems, this is one hypothesis that I would explore.

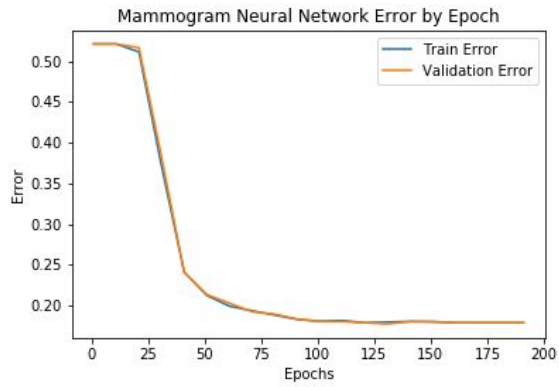
As discussed, I chose to focus my analysis on finding the optimal value of K. The largest opportunity for improving the KNN learner over these problems is to include the distance metric and distance weighting in a search for the optimal hyper-parameters. As an improvement I would input all 3 of these parameters into GridSearchCV to find the optimal combination of parameters in each problem.

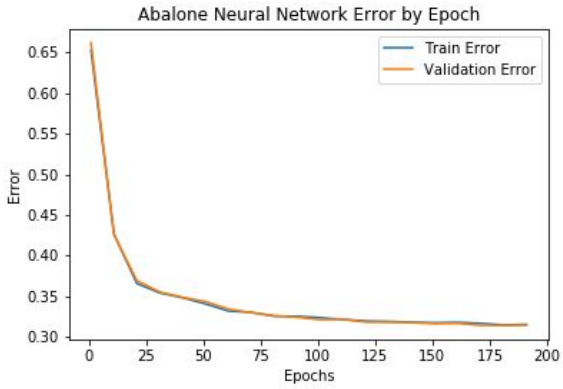
NEURAL NETWORK

The hyper-parameters that need to be considered when designing a Neural Network are: number of nodes and layers, learning rate and activation function. I chose to use 1 hidden layer as the majority of hypothesis can be reflected with 1 hidden layer. I used GridSearchCV to find the optimal values of the remaining hyper-parameters (code provided if interested).









Mammogram Mass

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Neural Network	82.5301	75.4491	0.272635	0.000197887

Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Neural Network	71.5055	71.0526	1.48332	0.00127101

The optimal hyper-parameters for the Mammogram problem are: {'alpha': 0.001, 'activation': 'logistic', 'hidden_layer_sizes': (8,)}. The optimal hyper-parameters for the Abalone problem are: {'alpha': 0.01, 'activation': 'relu', 'hidden_layer_sizes': (8,)}.

The first element of the Neural Network results that stand out to me is how NN applied to the Abalane problem seems to continue to be improving with additional examples even as we reach the threshold of examples available to our learner (4k). This shows that the NN is unable to fully capture the complexity of the hypothesis space with the examples available, and to converge to an optimal concept we would need additional examples. This contrasts with the other learners we've examined thus far which all "converge" to a minimum error at around 2k-2.5k examples. This makes intuitive sense - given the massive hypothesis space that NN represents, it takes more examples to narrow that space into an optimal hypothesis for any given problem.

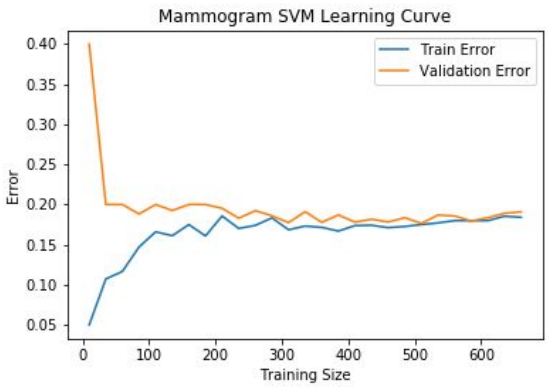
Comparing the Neural Network results between datasets, one can clearly see the effect of a differing alpha in the NN learning curves by epoch. Given the smaller learning rate for the mammogram problem, it takes more iterations or epochs to start converging on the optimal hypothesis. With a higher learning rate in the Abalone problem, the NN starts to converge much earlier.

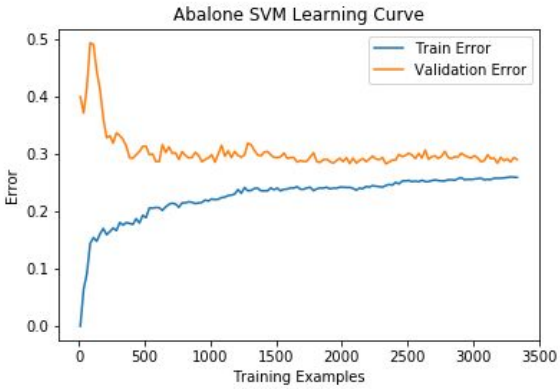
One way in which I would explore improvements to the Neural Network learners as applied to this problem is to tweak the range of values included in the GridSearch optimizer. For both problems, the optimal # of nodes in the hidden layer was 8, which is the highest option for # of nodes provided to GridSearch. It is possible that models with even larger number of nodes in the hidden layer would perform better than the final models here. With that said, we are already able to capture the interaction between all input features with a hidden layer of 8 nodes, so I would be surprised if increasing the # of nodes would have much improvement.

By a similar argument, I would experiment with the number of hidden layers in the neural networks as I fixed hidden layers at 1 for both models. Again, there aren't any interactions between features that are unable to be captured by the current model so adding more layers may simply cause overfitting and/or backpropagation to fall into local minima (which is why I narrowed to 1 hidden layer in the first place).

SUPPORT VECTOR MACHINE

The hyperparameters that need to be considered in a Support Vector Machine (SVM) learner are the kernel function, C (cost of misclassification, only relevant for some learners) and gamma () For both problems I created an SVM using two different kernels: linear and Radial Basis Function (RBF) Both of the below learning curves are for the SVM with an RBF Kernel.





Mammogram Mass

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
SVM (RBF)	81.7771	76.0479	0.0083859	0.004107
SVM (Linear)	81.4759	74.8503	0.0238202	0.00341296

Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
SVM (RBF)	73.6306	70.3349	0.513616	0.189712
SVM (Linear)	70.6076	69.378	1.63097	0.1317

The first interesting thing to note about our SVM model is the learning curve behavior as training size increases. Unlike many of the other models, training error actually grows as we increase the total number of training examples, while validation error decreases as we increase training size. This is particularly stark with training size <100, my hypothesis for this behavior is that since SVM is finding a “boundary” between TRUE and FALSE examples, it’s easier for the learner to split data evenly when there are fewer examples to draw from. As more examples are added, more noise is added as well, and the boundary between TRUE/FALSE becomes fuzzier. In a way, the SVM model is overfitting when we have fewer examples to train on. A growth in learning error with increasing Training Set size can be seen in both learning problems.

Related to the above, some of the specific qualities of an SVM model can be seen through the Abalone learning curve. The validation error in the Abalone learning curve converges by about 600 training examples, and doesn’t really improve with the addition of 3500 more examples. In comparison, all of the other algorithms took around 2000 examples to converge on the lowest

error. Given SVM only uses the vectors on the decision boundary(s) to determine the model representation, I would argue that in the Abalone dataset, enough of the support vectors on the decision boundary(s) are present by ~600 examples that SVM does not need to see more examples to determine it's representation. This trend is not as clear in the Mammogram problem as all of the Mammogram learners converge to the lowest error between 50-100 examples.

PERFORMANCE COMPARISON!

Mammogram Mass

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Decision Tree	81.9277	74.8503	0.00119901	0.000138998
Boosting (DT)	83.2831	76.6467	0.433827	0.0303068
KNN	82.6807	77.8443	0.00157309	0.0188339
Neural Network	82.5301	75.4491	0.380738	0.000198841
SVM (RBF)	81.7771	76.0479	0.00753689	0.00420499
SVM (Linear)	81.4759	74.8503	0.016531	0.001894

Abalone Age

Learner	Train Acc(%)	Test Acc(%)	Train Time(s)	Query Time(s)
Decision Tree	72.6729	70.933	0.0131099	0.000469923
Boosting (DT)	75.5462	71.0526	0.854718	0.042521
KNN	71.6851	70.0957	0.00434995	0.11816
Neural Network	71.5055	71.0526	2.36836	0.00162506
SVM (RBF)	73.6306	70.3349	0.650224	0.450793
SVM (Linear)	70.6675	69.2584	0.800089	0.117523

A few more trends show themselves when we look at the performance (accuracy, train time, query time) of our learners stacked up against each other. First off, Boosting has the best accuracy over the training set in both problems. However, Boosting only has the best accuracy over the test set in the Mammogram problem. This showcases the common characteristic of boosting algorithms to more closely fit the training set, without introducing overfitting or a lack of generalization when applied to unknown examples.

Boosting and Neural Networks consistently took the longest time to train in both problems. These are the only two "iterative" algorithms that we examined and it makes intuitive sense that they would have a higher train time than alternatives such as KNN or Decision Tree which train a model in a single pass and do not require multiple iterations. In fact, train time for KNN and Decision Tree learners were the lowest in both problems. Surprisingly, Decision Tree train time was less than KNN in the mammogram problem, theoretically the train time for KNN is negligible as the only action being done is storing each example in a database or dataset. On the other hand, Decision Tree stores all of the data points and then builds a model on top of these points. Given train time was lower for KNN than DT in the Abalone problem, my hypothesis is that the

sklearn KNN module is setting up some additional indexing for eventual querying that takes computational resources.

SVM train and query time seem to grow exponentially with additional data. The train and query times were ~100x longer for both kernels when moving from 860 examples (Mammogram Mass) to 4k (Abalone Age). This was a surprising result to me as I understand SVM only uses the support-vector, or the vectors closest to the decision boundary to determine the final model representation. I would expect that SVM train time grows linearly or less with respect to number of examples given the number of support-vectors should at most linearly. Given more time I would want to explore this more to understand what's driving this behavior.

For the Mammogram Mass problem, KNN had the highest accuracy over the test set at 77.8%. It also had the smallest margin between train and test error. Three other learners actually performed as well or better than KNN over the training set, while KNN still managed to have the best accuracy over the test set. This seems to imply that the KNN algorithm generalizes better than the others, at least for the Mammogram problem.

With the above in mind I would immediately say that KNN is the best "performing" algorithm for the Mammogram dataset, however it does have an increased computational cost at query time compared to the other algorithms. When deciding which algorithm to implement in a practical setting, we need to weigh the cost of increased query time (e.g. how often are queries run against the model and what computational resources are available) against how important it is that our model is correct. If we would rather trade off higher accuracy for decreased query time, then the SVM (RBF) model is a great option as it has lower query time and only slightly lower accuracy over the test set (0.6ppt lower). For diagnosing whether mammogram masses are benign or malignant, I would put the highest importance on model accuracy, and therefore select KNN as the "best" model.

For the Abalone Age problem, boosting and neural net algorithms had the highest accuracy over the test set at 71.05%. The neural net algorithm had much higher train time than boosting at 2.36 seconds opposed to 0.86. Even with a higher training time, I would advocate for using the neural network algorithm in practical applications for a few reasons. First, training occurs infrequently, so a higher train is acceptable given the low frequency of actually training a model. Second, the query time of NN is ~30x faster than that of Boosting. Since querying will theoretically occur much more often than training, it makes sense to place higher weight on these times.

SOURCES

Mammogram Mass Dataset, UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets/mammographic+mass

Abalone Dataset, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/abalone