Victor Ma
vma3@gatech.edu
Machine Learning HW #1

Datasets

1. The Ford alertness dataset[1], hosted on Kaggle.com. This dataset contain the results of experimental trials of hundreds of different drivers recorded at 100ms intervals (2 minutes total) during a driving simulation. Each observation consists of 30 different continuous-valued measurements of environmental, physiological, and vehicular factors. Additionally, there is a binary variable which denotes whether or not the driver is alert. In total there are ~604K instances in the training set, and ~120K instances in the test set.

2. The Covertype dataset[2] hosted at the UCI machine learning repository. This consists of over 560K observations of 7 different types of forest cover in the Roosevelt National Forest, Colorado. Each observation corresponds to a distinct 30x30 meter cell and includes 13 types of cartographic measurements, including position, shade, wilderness area, and soil type. They are a mixture of both continuous and categorical data (including one hot encoding of a 40-level category feature). The data is split into a training and test set at a ratio of 4:1.

Classification problems

We seek to attempt to two different classification problems.

1. The first experiment is a binary classification problem using the Ford Challenge dataset which consists of continuous valued features. The objective is to predict whether or not a driver is alert for a given set of measurements. Although this is data involves time series, this fact will be ignored and the challenge will be to make accurate predictions based on only features observed at single instances.

2. The second experiment is a 7-way classification problem using the Forest Cover type data set, which consists of continuous, discrete, and categorical data (encoded in dummy variables). The objective is to predict the type of forest cover exhibited by a certain 30x30m region in a national forest, given various cartographic information, including soil type.

We believe the first problem is interesting because it attempts to perform classification using time series data, i.e. data which inherently has some information not captured by instantaneous measurements, but without consideration of the time element. Therefore, the

[1] https://www.kaggle.com/c/stayalert
[2] https://archive.ics.uci.edu/ml/datasets/Covertype

question is how well can we do without temporal elements? Such a problem also illustrates a real benefit of machine learning, i.e. real-time detection of drivers' lack of alertness which is potentially dangerous on the road. The second problem is interesting because it is an opportunity to practice classification beyond the binary case which has been discussed in lecture and is used frequently for examples and learning. Furthermore, the Forest cover type data set consists of both continuous, discrete, and categorical data so we can see how various methods perform with it. For example, categorical data is no problem for random forests, but for neural networks we have to perform one hot encoding. Both problems involve very large data sets, though because of time constraints they are heavily sub-sampled for this report.

Software

All work was performed using Python and two machine learning modules: 1) scikit-learn for decision trees, boosting, K-nearest neighbors (KNN), and support vector machine (SVM) algorithms; 2) pybrain for artificial neural networks.

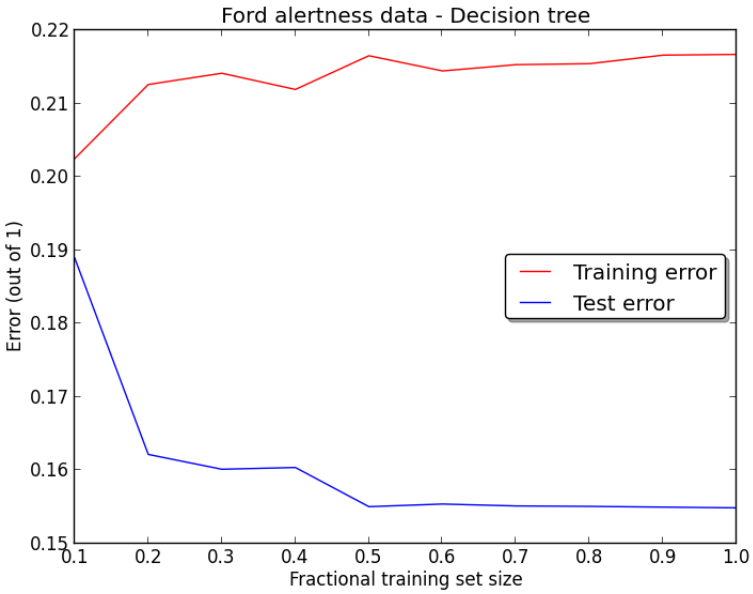
Ford alertness dataset

We chose to pre-process the data to improve run-times. This included feature selection and standardization/normalization of the data.

We assess classifier quality through mean accuracy using a traditional training and test data split and show its performance as a function training dataset size. The training set size is varied from 10% of the original size to 100% in 10% increments.

Decision tree

For the decision tree, we used the *Gini impurity* measure to assess the quality of any potential split, i.e. the feature and threshold which gives the best *Gini impurity* score is selected for the split. Although Sci-kit does not implement pruning, there are several possibilities for "pruning-like" procedures. Specifically, we restricted the maximum depth of the decision tree, and thus prevent the decision tree from growing too large and possibly overfitting based on unimportant features. By experimenting from a range from 1 to 22, we found that, surprisingly, a tree of depth 3 gave the best classification performance. Even more surprisingly a tree of depth 1 gave the next best performance. It is clear that the three features (two environmental and one vehicular) are highly indicative of whether the driver is alert, and that the other 19 features were (relatively) unimportant.

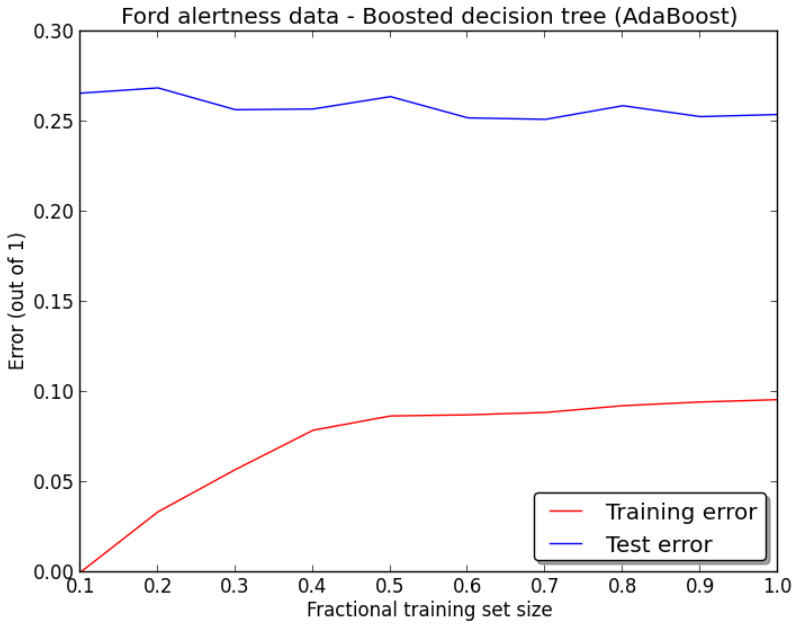


The average training and test error for the decision tree is plotted as a function of training data size. While we would typically expect training error to UNDER-estimate the test error, in this case, we actually see that training error starts out a bit higher than the test error. The difference grows even further as the training set size approaches 100% and the test error decreases. We suspect in that this is because the training set was a "lucky draw", and that it must have contained many hard-to-predict (for decision trees) instances compared to the test set. In other words, the test set could have had easier cases.

Boosted Decision Trees (AdaBoost)

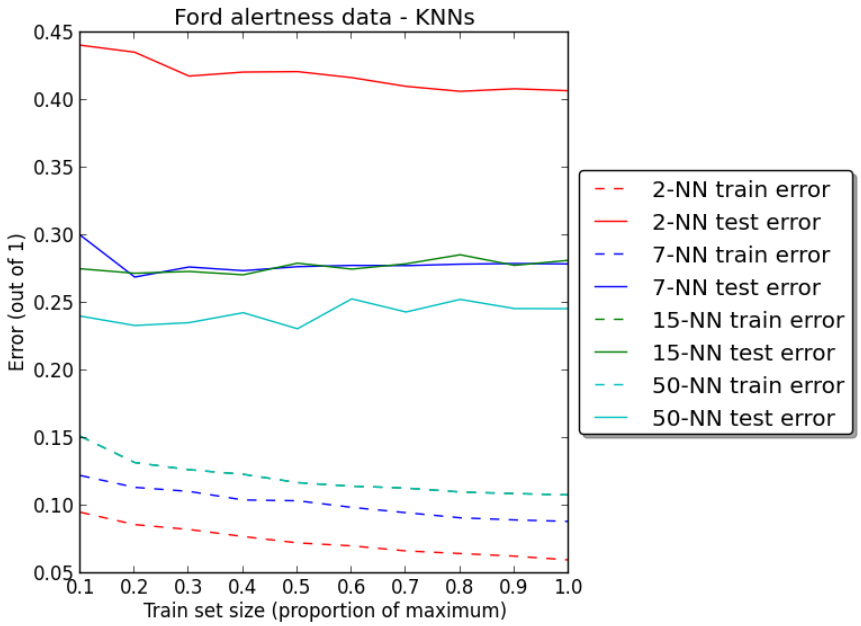
To test boosting, we used decision trees as a weak learner in the Adaboost algorithm. Specifically, we used set of 600 single node decision trees, i.e. max depth = 1 (less than the max depth before), with the same splitting criterion as before. As expected, the boosted decision tree's training error increases as a function of the training size, while the the test error decreases slightly. This time, the training error is indeed an underestimate of the test error by a factor of 1.5x. While with simple decision tree the error was less than 16%, the higher capacity boosted trees have an error of more than 25%. We learned that in practice, the boosting generally never overfits with more learners, and experiments varying the number of weak learners (not shown) seemed to confirm this. The reason that boosting performs

worse could be that the data is highly noisy and overlapping and it assigns too much weight



on difficult examples, which may not appear in the test set.

K-Nearest Neighbors

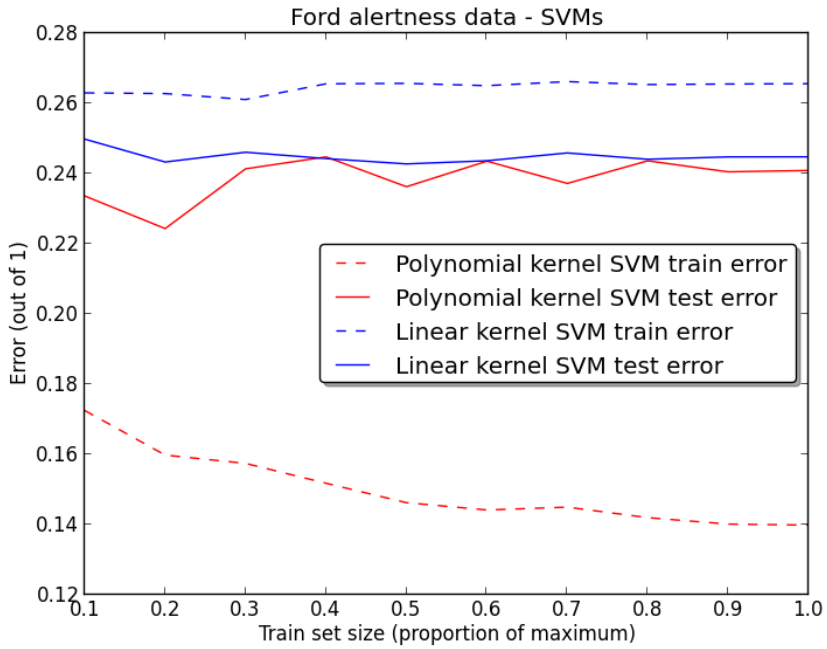


For K-NN, we used Manhattan distance with uniform weighting of K= 2, 7, 15, or 50 neighbors. For K=2, the test error is quite high (45% at 100% of max training set size) though

the training error is very low (6%). At K=7, we see the best test error decrease to 27% while training error increases to 9%. At K=15, we see a tipping point. The test error is stays roughly the same at 27%, and then at K=50 the test error actually increases from 24% to 25% as the training set size increases. Nonetheless, the highest K has the the lowest test error despite that fact that higher K suggests higher bias and lower variance.

SVM

For our SVM implementations, we use a regularization value of C=1 along with the polynomial and linear kernels. As with the K-NN, we see that for the polynomial kernel training error has an overall downward trend as the size of the training size increases. Meanwhile, the test error actually *increases*. This is a sign that the SVM has overfit the training data, and overfits it even more when there are more training examples. Perhaps the capacity of the polynomial kernel is too high, because we see the opposite happening with the linear kernel. That is, the test error decreases and the training error increases with the size of the training set. The true distributions of the different data classes (driver is alert, driver is NOT alert) may be somewhat linearly separable. In fact, this may be why the decision tree works well; decision trees impose

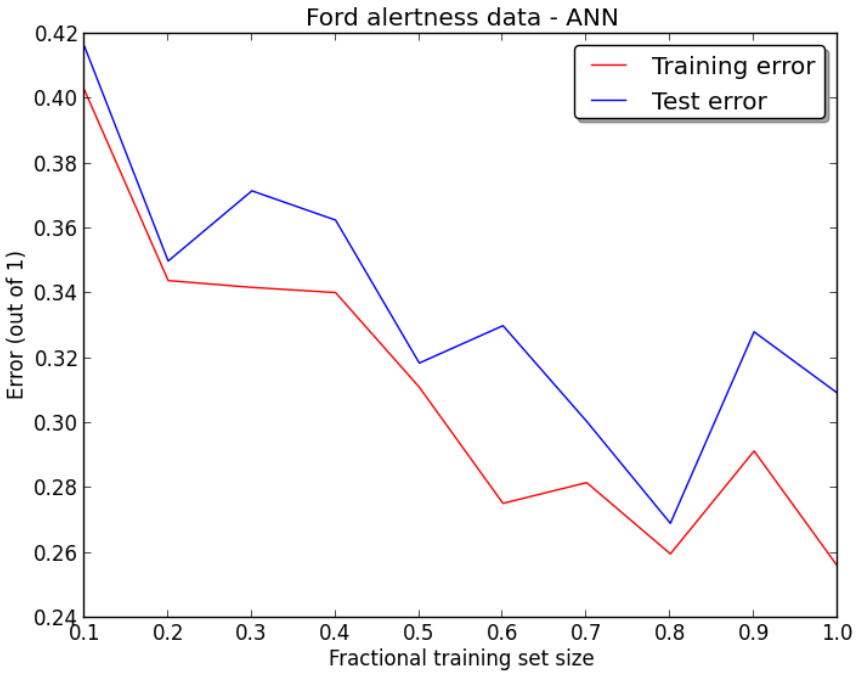


hyperplanes along the axes of the features.

ANN

We constructed a feed-forward neural network with two hidden layers. The first hidden layer consists of 15 sigmoid units, while the second has 7 tanh units. These were chosen somewhat arbitrarily. The neural network is trained using backpropagation and using only 4 epochs, as we found through experimentation with a validation set that it quickly began to overfit after that point. In the graph, we see that both training and test error drop sharply, from

40% and 42% respectively, down to 26% and 31% respectively. It is difficult to resolve performance of the ANN with that of the decision tree and Adaboost, as we would assume as the training set increases, it becomes increasingly hard to fit it.



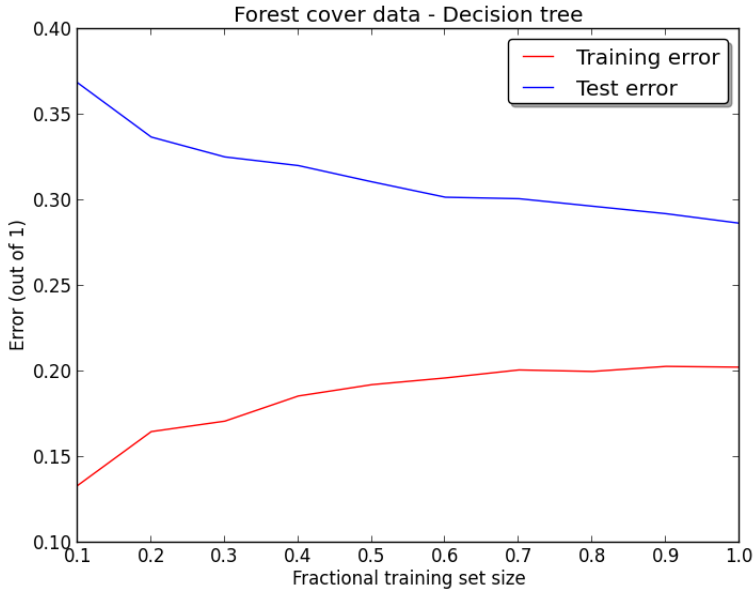
Forest Cover dataset

With the Forest Cover dataset, we only pre-processed the data by scaling to unit variance and subtracting the mean. Feature selection was unnecessary. As before, we assess the classifier quality via mean accuracy on a held out test set. Again, we show the performance of the classifier as a function of the training data set size. Given that this a 7-way classification problem (identify one of the 7 forest cover types) but with similar data set size, we expect this to be tougher in some regards than the Ford alertness classification.

Decision tree

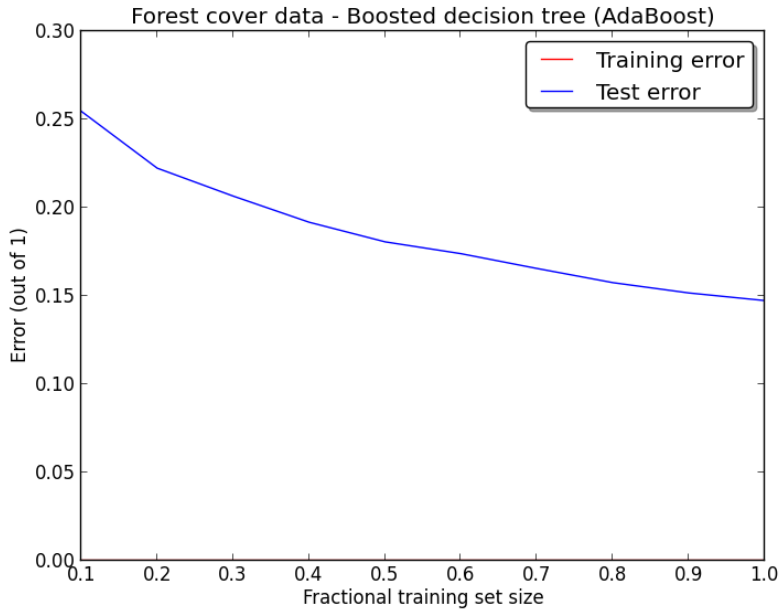
We used a decision tree with a maximum depth of 22, which we set according to prior experiments, in order to provide some pruning effect. The splitting criterion is again the Gini impurity measure. The decision tree exhibits expected behavior. The test error decreases from ~36% to ~29% from a training set proportion of 10% to the full data set. The training error increases from ~14% to ~20%. The the test error is much higher than it was for any of the classifiers in the Ford dataset. This seems reasonable, if only because this is a 7-way classification problem, and as such, there is about 7 times fewer samples per possible class.

Furthermore, there are fewer features, but two of the features (wilderness area and soil type)



are encoded in dummy variables. This increases the dimensionality of the data from 13 to 54.

Boosted decision trees (AdaBoost)

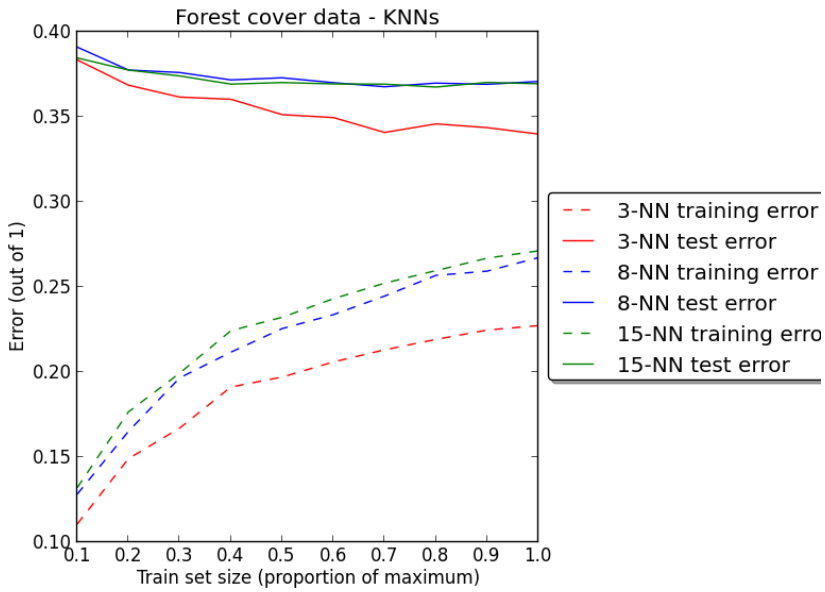


We use Adaboost with decision trees again as weak learners. This time, the maximum depth for the decision trees were set to 12 to combat overfitting in a pruning-like fashion. The number of learners used was 600. At each increment of training set size, AdaBoost is able to

fit the training set perfectly, but as the size increases, the test error decreases from ~25% to less than 15%, far better than vanilla decision trees.

K-NN

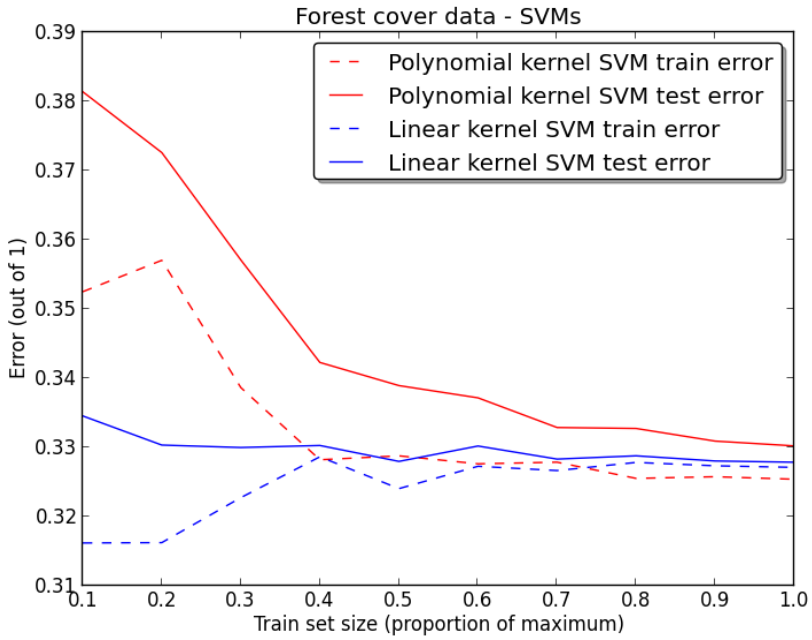
For K-NN, we used distance-based weighting for K = 3, 8, and 15 neighbors and a Euclidean distance metric. At K=3, the test error decreases from ~38% to ~34%, while the training error increases from ~12% to 23%. At K=8, both test and training error have similar shapes but are higher at full training size. There is almost no change between K=8 and K=15. Thus, the error is higher than with the decision tree, and far higher than the AdaBoost. The reason why the performance is not so good is probably because of two reasons: 1) different wilderness areas have different forest cover types, and are thus more discriminating, but do not receive more weight as a feature due to the K-NN assumptions; and 2) within these wilderness areas the forest covers can be mixed, so that closeness doesn't necessarily mean same forest cover. In fact, there may be discontinuities in that around the boundaries of wilderness areas, forest



regions may have very similar attributes but have distinctly different forest covers.

SVM

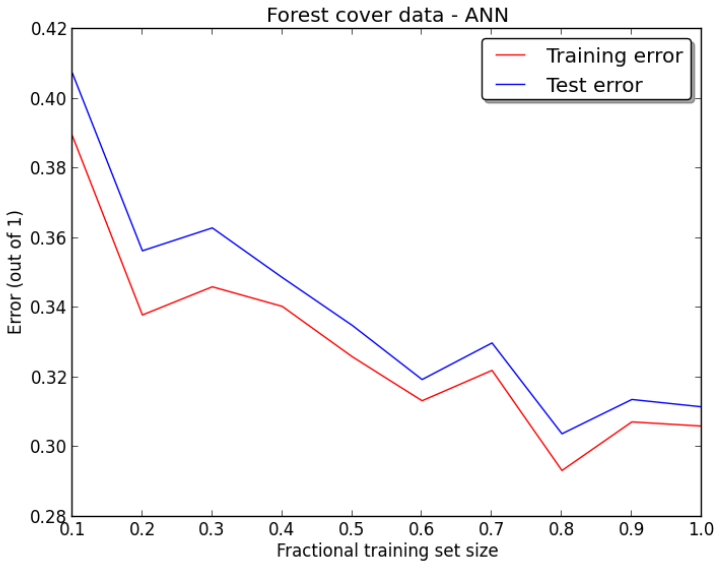
The SVMs used were again polynomial and linear kernels with a regularization parameter C=1. For the polynomial SVM we see that the training and test errors decrease alongside each other, with the test error being roughly a percentage point on above the training error. The training error is thus a good approximation of the test error for our data sets. That said, at 100% training set size, test error slightly above 33%.



For the linear SVM, the test error decreases slightly from ~33.5% to ~32.8%, while training error rises a bit from 33.2% up to ~32.7%. At above 50% of the maximum training set size, the training error is a tight approximation of the test error (though it's not bad at 10% size as well). For both SVMs, we wanted to experiment with other values of C to play with the bias-variance tradeoff, but we didn't due to the long computation times involved.

ANN

The artificial neural network was constructed with two hidden layers: the first layer of 30 sigmoid units, and the second layer of 15 tanh units. Training of the neural network stopped at 10 epochs, based on prior experiments on a validation set (not shown). Both training and test error are shown to decrease, going from 39% and 41% respectively, to ~31%. This is similar to what we saw before with the Ford alertness data set.



Summary of results

By comparing the different classifiers' performances on the Ford alertness and Forest Cover type data sets, along with experimentation with the classifier parameters, we observed how much they can vary, and how big a difference there can be in mean accuracy. For example, in the Ford dataset, even though most more complex classifiers were getting around 25% error, a simple decision was able to attain 15.5% error. Meanwhile, for the Forest Cover data set, the AdaBoost won the day with 14.7% error.

Although this isn't representative of all case, I observed that the decision tree was the fastest to train, and that AdaBoost didn't take as long as the others either. This training speed and the fact that I needed to work less to tune their parameters meant that, in the future, I will attempt to use these types of classifiers first. I will either establish a baseline, or get a decent performance.

Unfortunately, in retrospect, I should have chosen to use smaller datasets as it would have been more conducive to a deeper exploration of the different classification algorithms, although I can say that practical experience with large datasets is useful.

	Ford Alertness classification		**Forest Cover type classification**	
Classifiers	**Training error**	**Test error**	**Training error**	**Test error**
Decision tree	21.7%	**15.5%**	20.3%	28.7%
AdaBoost	**9.6%**	25.4%	**0.00%**	**14.7%**
K-NN	13.2%	24.4%	22.8%	34.0%
SVM	14.0%	24.1%	32.7%	32.8%

ANN	25.6%	30.9%	30.6%	31.2%

1. This table shows the training and test errors on the full (100%) training set considered, for each classifier using the best parameters found during experimentation.
2. The **bold** numbers represent the best error in each column.

	Ford Alertness data set (20K training instances, 22 features)	**Forest Cover data set (20K training instances, 15 features)**
Classifiers	**Training time (sec)**	**Training time (sec)**
Decision tree	1.38	0.32
AdaBoost	164.62	98.68
K-NN	491.55	38.64
SVM	1864.22	40.70
ANN	~500	45.36

1. This table shows the training times on the full (100%) training set considered, for each classifier using the best parameters found during experimentation. For comparison, different data set sizes are used.
2. These times come from a single trial, run a computing cluster.