ECS 171 Final Report

Aaun Abbas
Hilal Alsibai
Christopher Chen
Miguel Covarrubias
Jesse Dyer
Pei Guo
Alex Kot
Raymond Lau
Kent Wang
Ian Woods

December 14, 2014

1 Abstract

In this project, we compared the effectiveness of several machine learning classifiers at determining forest cover types from cartographic variables. The classifiers we used were artificial neural networks, k nearest neighbors, and random forests. Our results showed that random forests more accurately predicted the correct forest cover type than any other classifier.

2 Introduction

Determining forest cover type from purely cartographic variables is important in situations where it is unfeasible or impossible to obtain cover type data through empirical methods. Thus, being able to accurately estimate the cover type of an area is of great interest to forest scientists [1]. At the same time, the dataset provides a large amount of well structured data (581012 samples with 54 attributes) upon which different machine learning techniques can be tested and compared, making it an attractive dataset for computer scientists [2, 3, 4, 5, 6, 7].

3 Methods

We use three different classifiers to determine forest cover type: artificial neural networks, k nearest neighbors, and random forests.

3.1 Artificial Neural Networks

3.2 k Nearest Neighbors

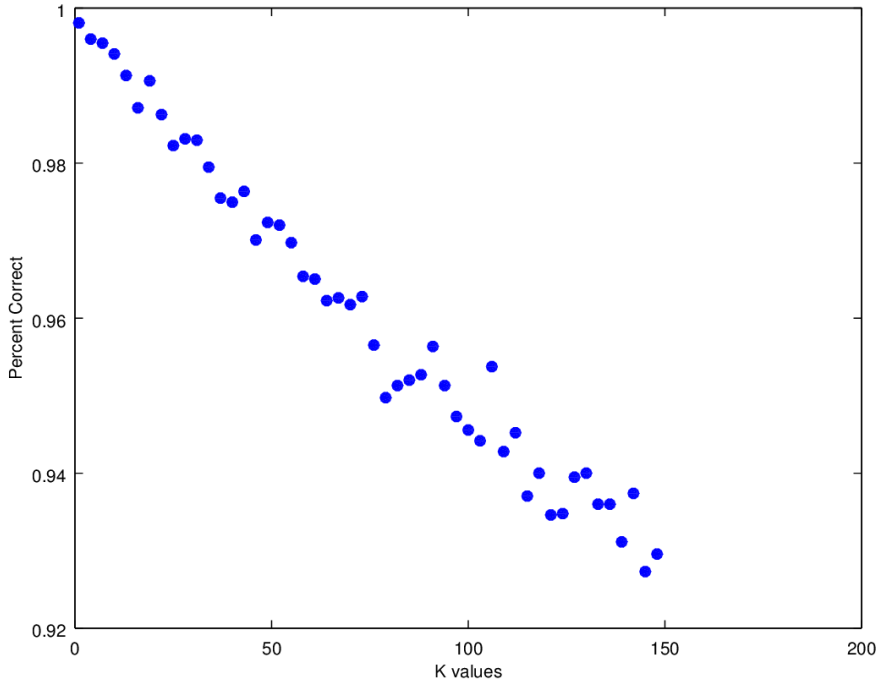
The k-Nearest Neighbors (KNN) Algorithm involves classifying a given sample as a particular type, given the type of the k closest samples to it, as determined by some distance function. Our initial approach was to use a standard Euclidean distance function, defined as follows:

$$\sqrt{\sum_{k=1}^{n}(x_{jk} - x_{ik})^2}$$

We decided to begin with this naive approach to isolate failure points (essentially, if something were to work incorrectly, we could rule out the distance function as failure point). We debated creating a distance function tailored to the dataset by hand, or by simply trying many different possible distance functions. However, we decided to generate a distance function with the largest-margin nearest neighbors algorithm (more precisely, we generated a

transformation matrix that was used to transform the data, and still used the Euclidean distance).
Using different values for k can theoretically alter the predicted results - for example, if a sample of unknown type is closest to a sample of type A with a calculated distance of 1, but two samples of type B are at a distance of 1.1 from our unknown sample, choosing $k = 1$ will yield the result A for the unknown's type, and $k = 3$ will yield the result B. We decided to test a wide range of k values.

3.2.1 Results of KNN Using Various K Values

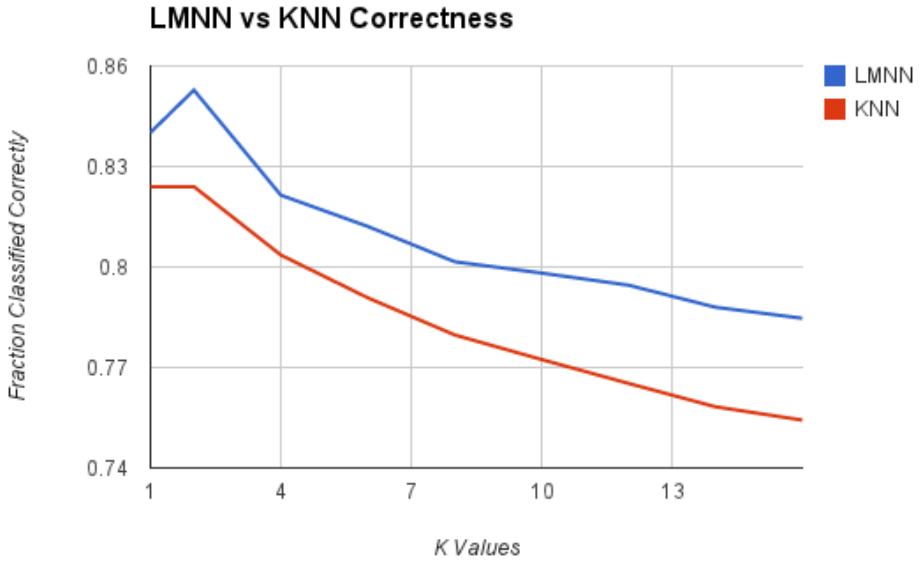


As can be seen in the above graph, increasing the value of k reduced the accuracy of the classifier. We tested over a range from 1 to 150 with a step size of 3 (to reduce computation time while still revealing the overall trend). This makes intuitive sense - imagine a KNN implementation with $k =$ the size of the data set. Say there are n samples of the most common type, and z total samples. Every sample would be labeled as the most common sample, resulting in an accuracy of $\frac{n}{z}$.

3.2.2 Brief Description of LMNN

LMNN (Largest Margin Nearest Neighbors) is a variant of KNN that can be implemented in two ways. One method is to learn a Mahalanobis distance function, which measures the distance (in standard deviations) from the center of a cluster of similarly typed samples. Another method, and the method our implementation used, is to learn a transformation matrix that, when applied to the dataset, yields higher correct classification rates [**?**]. It learns this matrix by iteratively applying small perturbations to the transformation matrix with the goal of maintaining a large distance between "impostors" (close neighbors that are of the wrong type) and the perimeters that surround groups of similarly-typed neighbors. LMNN is very helpful because it saves use the trouble of testing many different distance functions, or trying to derive a distance function by hand for each particular dataset. Pictured below are the results of our LMNN classifier, compared to the standard KNN classifier.

Results of LMNN



LMNN was able to boost the correct classification percentage in every case. On average, we saw 2.46% better classification rates. It took quite a

lot of extra time to learn the distance function, though - running the LMNN algorithm on the entire UCI dataset took on the order of one hour on a relatively high-powered desktop computer (quad core Intel i5 processor at 3.4GHz, all CPUs at 95% + load for most of the computation).

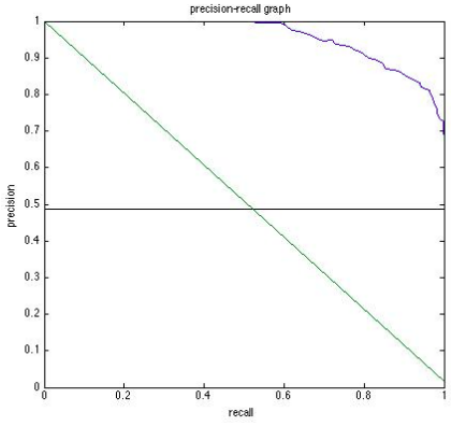


Figure 1: PR Curve for class Aspen

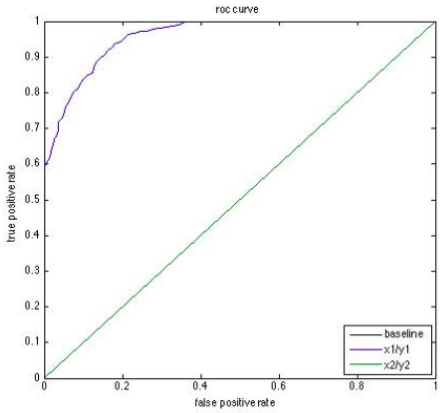


Figure 2: ROC Curve for class Aspen

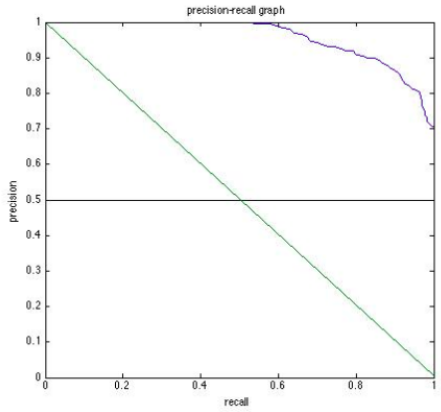


Figure 3: PR Curve for class Cottonwood

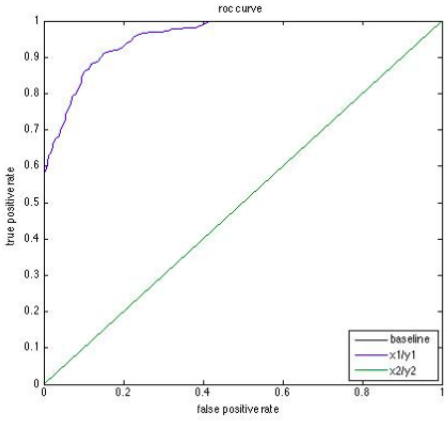


Figure 4: ROC Curve for class Cottonwood

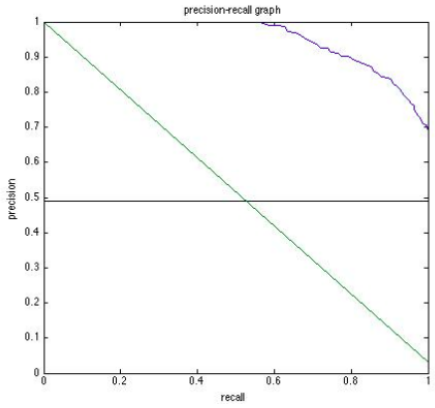


Figure 5: PR Curve for class Douglas Fir

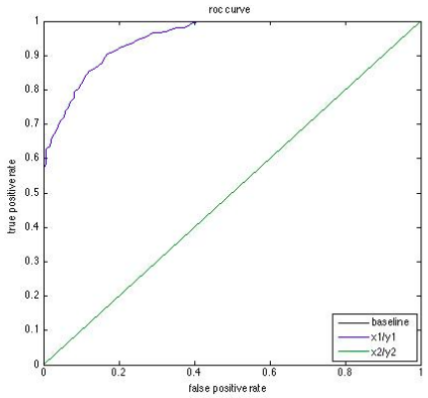


Figure 6: ROC Curve for class Douglas Fir

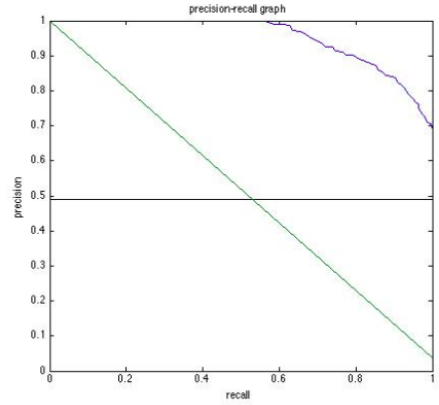


Figure 7: PR Curve for class Krummholz

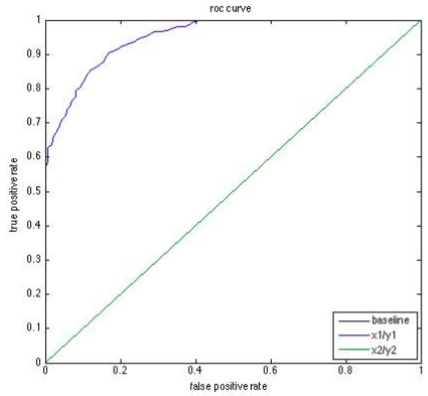


Figure 8: ROC Curve for class Krummholz

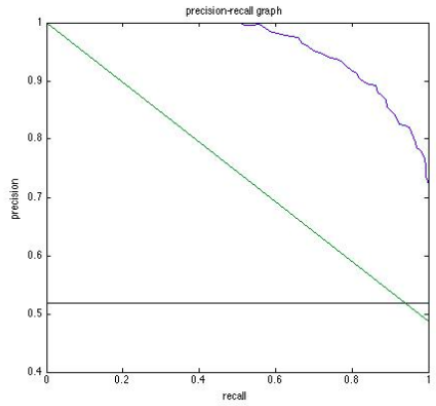


Figure 9: PR Curve for class Lodgepole Pine

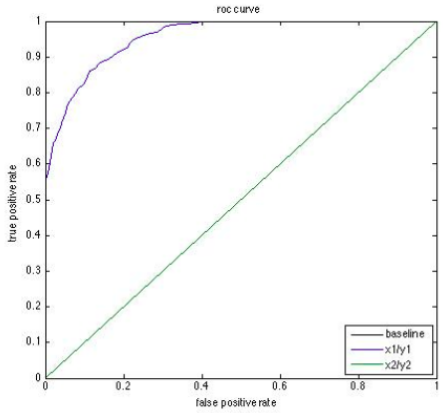


Figure 10: ROC Curve for class Lodgepole Pine

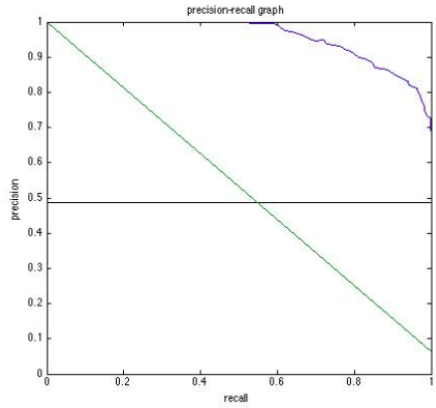


Figure 11: PR Curve for class Ponderosa Pine

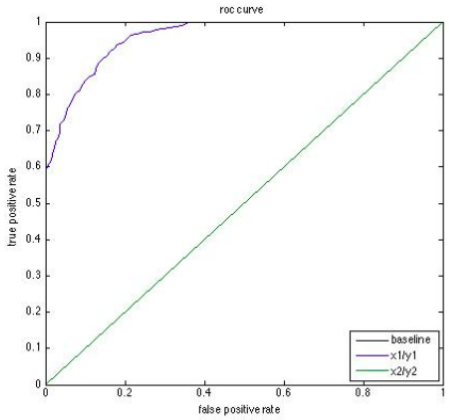


Figure 12: ROC Curve for class Ponderosa Pine

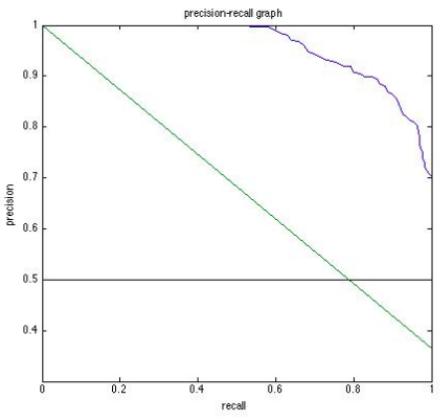


Figure 13: PR Curve for class Spruce Fir

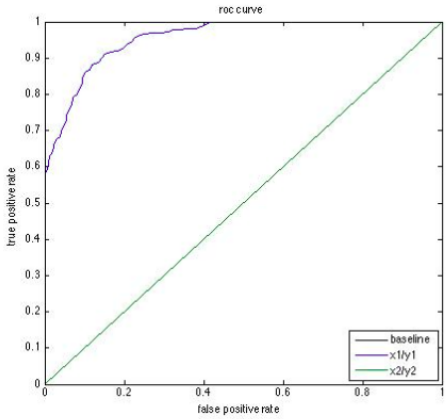


Figure 14: ROC Curve for class Spruce Fir

3.2.3 Final KNN and LMNN Analysis

Cover Type	AUC
Aspen	.892
Cottonwood	.877
Douglas Fir	.885
Krummholz	.888
Lodgepole Pine	.891
Ponderosa Pine	.883
Spruce Fir	.873

From these graphs it can be seen that the performance from each of these classifiers is far greater than a random guess, and that they were also relatively consistent across the cover types.

3.3 Random Forests

We used Liaw and Wiener's R port [8] of Breiman's random forest algorithm [9]. Before training the classifier on the samples, we preprocessed the data so that the 4 boolean categorical variables that were related to wilderness area were rolled up into a single variable. We did the same for the 40 variables (also boolean categorical) that were related to soil type. This was done in order to speed up the training of the classifier.

The random forest algorithm takes several parameters which control its

speed and performance. We used empirical analysis to determine optimal values for these parameters. The first of these was $mtry$, which controls the number of variables that are sampled at each split when building the decision trees. 7 was determined to be the optimal value, as increasing the value past 7 caused almost no change in error, but consumed far more system resources. The other parameter was $ntree$ which controls the number of trees to be generated in the forest. For the same reasons as $mtry$, 40 was found to be the optimal value.

To determine error, we used the OOB (out of bag) error [10]. We used this estimate because it is automatically generated by the algorithm while it runs, and because cross validation is not required when using OOB error, due to the way it is computed [9, 10], which saved us computation time. It has also been shown to be close to an optimal estimate of the generalization error[10].

4 Results

This should be an ROC curve.

5 Discussion

Why is one classifier better than another?

6 Conclusion

In this project we used 3 different classifiers to predict forest cover type based on cartographic variables: artificial neural networks, k nearest neighbors, and random forests. We determined that random forests outperformed the other 2 classifiers. In the future, we may try to examine the performance of these classifiers on other datasets and then compare those results to those given here, or we may use different classifiers on the cover type dataset to see if we can improve our classification accuracy.

7 Contributions

Aaun Abbas and Raymond Lau wrote all of the code that was related to random forests. Christopher Chen wrote all parts of the report that were related to random forests.

References

[1] Blackard, Jock A. and Denis J. Dean. 2000. "Comparative Accuracies of Artificial Neural Networks and Discriminant Analysis in Predicting Forest Cover Types from Cartographic Variables." Computers and Electronics in Agriculture 24(3):131-151.

[2] Joao Gama and Ricardo Rocha and Pedro Medas. Accurate decision trees for mining high-speed data streams. KDD. 2003.

[3] Nikunj C. Oza and Stuart J. Russell. Experimental comparisons of online and batch versions of bagging and boosting. KDD. 2001.

[4] Chris Giannella and Bassem Sayrafi. An Information Theoretic Histogram for Single Dimensional Selectivity Estimation. Department of Computer Science, Indiana University Bloomington.

[5] Johannes Furnkranz. Round Robin Rule Learning. Austrian Research Institute for Artificial Intelligence.

[6] Zoran Obradovic and Slobodan Vucetic. Challenges in Scientific Data Mining: Heterogeneous, Biased, and Large Samples. Center for Information Science and Technology Temple University.

[7] Arto Klami and Samuel Kaski and Ty n ohjaaja and Janne Sinkkonen. HELSINKI UNIVERSITY OF TECHNOLOGY Department of Engineering Physics and Mathematics Arto Klami Regularized Discriminative Clustering. Regularized Discriminative Clustering.

[8] A. Liaw and M. Wiener (2002). Classification and Regression by randomForest. R News 2(3), 18–22.

[9] Leo Breiman. Random forests. Machine learning 45 (1), 5-32. 2001.

[10] Breiman, Leo. Out-of-bag estimation. Technical report, Statistics Department, University of California Berkeley, Berkeley CA 94708, 1996b. 33, 34, 1996.

[11] Do, Huyen, et al. A metric learning perspective of SVM: on the relation of LMNN and SVM.