Classifying Forest Cover Type using Cartographic Features

Kevin Crain: craink@cs.stanford.edu Graham Davis: gbdavis@stanford.edu

Stanford University - CS 229: Machine Learning - December 2014

1 Introduction

Given elevation, hydrologic, soil, and sunlight data can we predict what type of tree would be in a small patch of forest? Our project attempts to predict the predominant type of tree in sections of wooded area. Understanding forest composition is a valuable aspect of managing the health and vitality of our wilderness areas. Classifying cover type can help further research regarding forest fire susceptibility, the spread of the Mountain Pine Beetle infestion[1], and de/reforestation concerns. Forest cover type data is often collected by hand or computed using remote sensing techniques, e.g. satellite imagery. Such processes are both time and resource intensive [2]. In this report, we aim to predict forest cover type using cartographic data and a variety of classification algorithms.

1.1 Data and Features

The data used in this report was taken from the UCI Machine Learning Repository [3]. The data set consisted of 15,120 samples of 30m x 30m patches of forest located in northern Colorado's Roosevelt National Forest. Certain attributes lend themselves well to human interpretation. For example, the correlation between aspect (compass direction of slope face) and sunlight intensity makes intuitive sense, seeing as some trees may thrive with heavy morning sun. Each data sample includes 54 attributes: elevation (in meters), slope, aspect (compass direction of slope face), vertical distance from water, horizontal distance from water, sunlight intensity at 9am, 12pm, and 3pm, 4 binary wilderness area designators, and 40 binary soil type designators. In some tests, the 44 binary attributes were ommitted. Each sample was classified into one of seven forest cover types: Spruce/Fir, Lodgepole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas Fir, or Krummholz. There are 2,160 samples classified as each of the seven cover types.

2 Models & Methods

To take measures against overfitting our models we employed 10-fold cross validation. Our entire data set of 15,120 samples was split into 10 evenly-sized groups $k_1, ..., k_{10}$. For each group k_i, we would train our models by conglomerating the other 9 groups before testing on k_i. We then average the percent error on each testing set k_i to determine the perecent error for the model.

Additionally, for some of our testing we removed the 44 binary features (labeled "boolean" on some figures) in an effort to see if shrinking our feature space's dimensionality would reduce overfitting and increase testing performance. Moreover, we knew that the models would run more quickly with only 10 features.

2.1 Principle Component Analysis

Principle Component Analysis (PCA) is a method of reducing the dimensionality of data while, usually, maintaining most of the data's variance. All dimensions of the reduced data are linearly uncorrelated, meaning the original data is projected into a space where each component is orthogonal to the others. Before computing the principal components, we zeroed out the mean of the data and set each coordinate of the data to have unit varaince so as to treat each attribute evenly. Computing the first principal component of the data can be formulated as:

$$\underset{\|u\|=1}{\operatorname{argmax}}\ u^T[\tfrac{1}{m}\textstyle\sum_{i=1}^{m} x^{(i)}x^{(i)^T}]u$$

Equivalently, this is the top eigenvector of the covariance matrix, Σ, of the data ($\Sigma = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)(x^{(i)} - \mu)^T$, where μ is the mean). Now, reducing the data to k-dimensions requires finding the top k eigenvectors of Σ (denoted $u_1, ..., u_k$). Transforming a data sample into the k-dimensional space now can be done by easily. Set the j-th element of the k-dimensional tranformed sample to be $u_j^T x^{(i)}$.

We used PCA to visualize our data in three dimensions (see Fig. 1). Clearly, the data is stratified and looks reasonably seperable. Additionally, we decided to run our multi-class SVM on the PCA-transformed data in various dimensions. Reducing the dimensionality of our data enabled our multi-class SVM to run more quickly. However, since variance is lost in reducing the data, we expected our results to suffer (see Fig. 2 in Results).

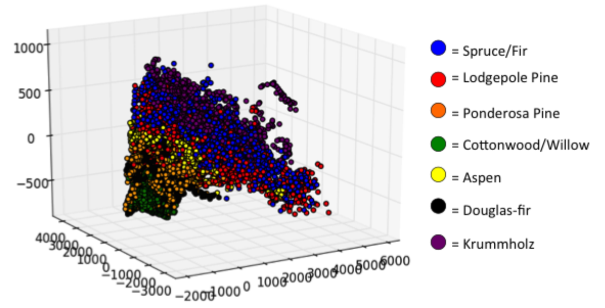

Figure 1: Three-dimensional representation of 8,000 samples via PCA

2.2 Multi-Class Support Vector Machine

The multi-class Support Vector Machine implemented in this report was imported from the sklearn library. This SVM uses the "one vs. one" approach to multi-class classification. With 7 cover types to classify, our SVM implements and trains $7 \times (7-1) \times 2 = 21$ seperate binary classifiers. Each binary classifier is trained using strictly examples of two cover types, meaning it always predicts one of two cover types. During testing, a test sample t is applied to each of the 21 classifiers. Each binary classifier "votes" for the cover type it predicts. Test sample t receives the label c, where c is the cover type receiving the highest number of "votes".

The binary classifiers within our mutli-class SVM run as normal SVM classifiers. Solving the ℓ_1 regularization dual problem provides an optimal value for α, which is used in the following equation, where the classifier predicts y = 1 if:

$$\sum_{i=1}^{m} \alpha_i y^{(i)} \langle x^{(i)}, x\rangle + b > 0.$$

Our multi-class SVM made use of the following "Gaussian" or Radial Basis Function (RBF) Kernel:

$$exp[-\gamma \|x^{(i)} - x\|^2]$$

To improve the accuracy of the multi-class SVM with respect to our data set, we sought to optimize two hyperparameters of our model: C (the penalty parameter present in the ℓ_1 regularization dual problem) and γ (our RBF Kernel coefficient). To do so, we used both grid search over the hyperparameters and 10-fold cross-validation. Using 10-fold cross-validation and multiple iterations, we settled on C = 8,500 and γ = 0.02 as the optimized values of our SVM's hyperparameters.

2.3 K-Means Clustering

As demonstrated in Fig. 1, the data obviously groups into cohesive clusters. So, we also decided to run an unsupervised clustering algorithm on the data. Since K-Means is unsupervised, it runs without observing the labels of the data. Upon convergence of the algorithm we observed each of the k clusters, labeling the cluster based on the most common cover type among the samples assigned to it. To increase the accuracy of our approach, for each value of k we run the algorithm 10 times. In each run we keep track of the cumulative sum of geometric distance from each sample to its respective cluster (i.e. "inertia"). Finally, we choose the cluster centers from the run which minimized this cumulative distance over the training dataset.

3 Results & Discussion

3.1 Principal Component Analysis

We knew reducing the dimensionality of the data would reduce the run-time of our multi-class SVM. However, the loss of variance would simultaenesouly decrease performance. Indeed, Fig. 2 confirms our expectations and graphically demonstrates the amount of variance captured by each principal component (see Table 1). When only using the first principal component the SVM performs better than our naive baseline, which always guesses the most common label from the training set. Within the first three principal components 98.35% of the data's variance is captured. The graph's curve shows the diminishing returns of running the SVM with more principal components. As expected, running the SVM with 10 dimensional PCA-transformed data (capturing 99.997% of the variance) performed only minimally worse than using the entire data set with all 54 dimensions. Also, the training error and generalization error were nearly identical for all runs.

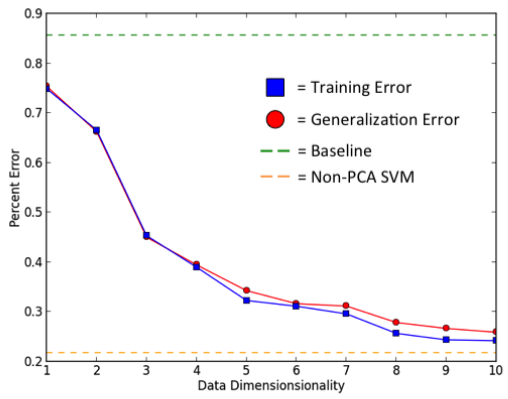

Figure 2: Training and generalization error of our SVM run on variable dimensions of PCA data

1	2	3	4	5	6	7	8	9	10
72.17	22.59	3.59	1.11	0.41	0.065	0.043	0.018	0.0012	0.00008

Table 1: Maintained percent variance for each of the ten principle components

3.2 Multi-Class SVM

As shown in Fig. 3, we ran diagnostics for the multi-class SVM on both the data with boolean information and without. The blue and red curves (with booleans) clearly demonstrate that as the number of training samples increases we see convergence between the training error and generalization error. Both the shapes and the relatively small gap between the curves indicate that high variance is not an issue. Similar findings occured when not using the booleans; in fact the difference between the purple and orange curves is always smaller than that between the blue and red. Therefore, we conclude that not using the booleans indeed lowers variance and reduces overfitting, comparatively.

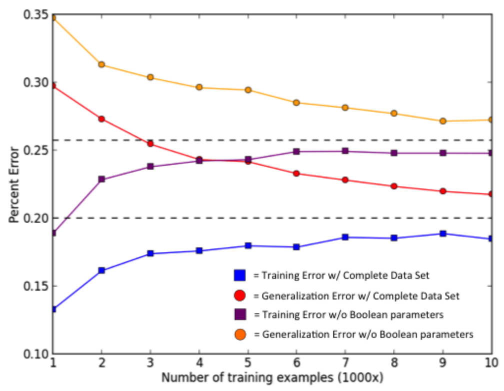

Figure 3: Training and generalization error of our SVM run with variable amounts of training data

For each sample exactly two of the 44 soil/wilderness indicators are turned on. Earlier, we hypothesized that the small amount of information contained in the 44 booleans could possibly hurt our model since 54 dimensions would be more prone to overfitting than 10. The results show that while reducing overfitting, the removal of the booleans also decreases performance on both testing and training. Essentially, the 44 boolean indicators provide valuable information at the cost of increasing the dimensionality of the data by more than a factor of 5.

When performing 10-fold cross validation on the entire data set with hyperparameters $C = 8,500$ and $\gamma = .02$, the generlization error was 21.36%, meaning our accuracy was 78.64%. When training this model on a random 70/30 split of the entire data set we obtained 81.35% training accuracy and 78.24% testing accuracy. When removing the boolean features these accuracies dropped to 75.21% and 72.75%, respectively.

3.3 K-Means Cluster

Initially, we ran K-Means with $k = 7$ clusters for each of the 7 covertypes. The results were poor, so we increased the number of clusters to allow for noise. That is, we knew from PCA's 3-D visualization that samples often "drifted" away from their cover type's average. We hypothesized that adding clusters would allow for each covertype to have multiple clusters, essentially making allowance for the "drifting." As seen in Fig. 4, additional clusters provided benefit up to a point. The curve indicates the diminishing returns of continuing this strategy, which also greatly increased the algorithm's runtime.

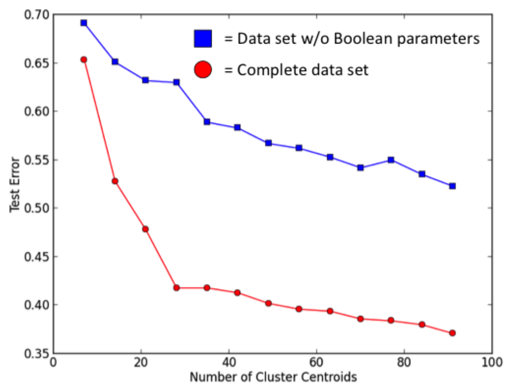

Figure 4: Test error of K-Means Clustering when run with varying numbers of clusters

4 Conclusions

Our multi-class SVM classifier and K-Means clustering algorithms performed quite well on both training and test data. In particular, our SVM classifier outperformed models used in studies involving a similar data set. For instance, one 1999 study by Blackard and Dean [2] reported 70.58% accuracy when classifying forest cover type using an artificial neural network. Furthermore, the same study reported 58.38% classification accuracy using discriminant analysis. These results demonstrate that an optimized, "one vs. one" multi-class SVM is a very accurate model with regards to forest type classification. Similarly, our SVM performed similarly well on both training and test data (Fig. 3). Thus, we were pleased to see that our model was not struggling with overfitting and was demonstrating lower than anticipated generalization error ($\approx 21\%$).

5 Future Work

Our main goal for future research is to approach our data set with other classification algorithms. First, we want to implement a "one vs. all" multi-class SVM. Because it seems to remain a relatively open question in machine learning research [4], we hope to gain insights into the relative performance of "one vs. one" and "one vs. all" SVM algorithms. Similarly, we would like to implement multinomial logistic regression and a neural network (similar to those implemented in Blackard (1999)[2]).

References

[1] D.A. Leatherman, Colorado State Forest Service entomologist (retired); 2/99. http://www.ext.colostate.edu/pubs/insect/05528.html Revised 9/11.

[2] Blackard, Jock A., Dean, Denis J. "Comparative accuracies of artifical neural networks and discriminant analysis in predicting foorest cover types from cartogrpahic variables". Computers and Electronics in Agriculture (1999).

[3] Bache, K. Lichman, M. UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science (2013).

[4] Milgram, Jonathan, et al. "'One Against One' or 'One Against All': Which One is Better for Handwriting Recognition with SVMs?"HAL (2006)