Leaf Classification

Machine Learning- Final Project Report

Harshal C. Mehta
Hcm140230@utdallas.edu

Shivika Prasanna
Sxp162630@utdallas.edu

Shreya Vishwanath Rao
Sxr169330@utdallas.edu

1. Introduction

There are about one million species of plants in the world. Classification of leaf species has been problematic for many years now, especially when the same leaf species are counted and identified more than once, creating duplicates entries. Automating plant recognition helps biologists in tracking and preserving species population, research on plant based medicines and crop and food supply management. The leaf species are considered because of their volume, prevalence and unique characteristics that help differentiate leaves of one plant from another. These leaf species that form the dataset have been collected by James Cope, Thibaut Beghin, Paolo Remagnino and Sarah Barman of the Royal Botanic Gardens, Kew, UK.

Extracted features of binary leaf images to identify 99 species have been taken. These features include margin, shape and texture. The input consists of 193 attributes and a class which include ID, 64 margins, 64 shape and 64 texture. The output gives the accuracy of the classifier used. This performance metric is used to determine the best classifier amongst those used.

2. Related Work

Classifying plant species based on leaf samples is a challenging task. Charles Mallah et al. [1] have worked on 16 samples, each of 100 plant species and have described a method that can work on smaller training sets and incomplete extraction of features.

They have designed a processing of 3 feature types (margin, shape and texture) along with a probabilistic framework. Texture and margin use histogram accumulation and a normalized description of contour is used for shape.

The density estimators have achieved 96% mean accuracy on a training set of 15 samples with unseen cross validation and 91% accuracy with only 4 training samples.

3. Dataset

The dataset consists of 1584 images of leaf specimen. These images have been converted to white colored leaves of binary format against a black background (Fig. 1). Each image has 3 sets of features which are shape contiguous descriptor for shape feature, interior texture histogram for texture feature and fine-scale margin histogram for margin feature. Each of these features consists of a 64- attribute vector per leaf sample.

Each instance in the dataset has a unique ID. The number of features are 3 sets * 64 attribute vectors, a total of 192 features. The number of total instances are 16 samples of 99 species, a total of 1584 species. The number of training instances are 990 instances and the number of test instances are 594 instances.

The margin feature is divided into 64 attribute vectors, namely, margin1, margin2, …, margin64. Similarly, the shape feature is represented by shape1, shape2, …, shape64. This is followed by texture feature which is denoted by texture1, texture2, …, texture64.

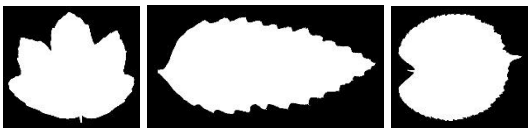

Fig. 1: Binary white leaves against black background

4. Pre-processing

The original dataset consisted of raw data that represent the extracted features of binary leaves. This dataset contained the class label in the second column. To simplify the dataset, we moved it to the last column. This was followed by checking for missing attributes. This was done by using a correlation matrix. A correlation matrix is used to obtain the dependencies between multiple variables at the

same time. Thus, we eliminated columns with lesser significance as compared to the overall dataset.

Many classifiers require dataset to be scaled. Scaling was done to standardize the array of independent variables or features of data.

Further, the data was split into training and testing data set that contained data in the ratio 80:20. This split was done to use the testing data to determine accuracy.

5. Classifiers

To analyze the samples, we used five different classifiers: Bagging, Random Forest, k-Nearest Neighbor, Support Vector Machine and Naïve Bayes.

5.1 Bagging

Bagging (Bootstrap Aggregation) is a machine learning ensemble method that gives improved stability and accuracy of algorithms used in classification and regression. It decreases the variance of prediction by generating additional data for training using combinations for repetitions to produce multisets of same size as the original dataset. This algorithm is applied to our dataset by using the "train" function that is provided by the "caret" package in R.

5.2 Random Forest

Random Forest is an ensemble learning method used for classification and regression. Random Forest train on different parts of the same training sets. They aim at reducing the variance. Random Forest differs from Bagging by picking a random subset of features. This algorithm is applied to our dataset by using the "train" function that is provided by the "caret" package in R.

5.3 k-Nearest Neighbor

k-Nearest Neighbor (k-NN) is an instance based learning method that is used for regression and classification. It uses the entire dataset as its model for representation and prediction. Hence, no learning is required. This algorithm is applied to our dataset by using the "knn3" function that is provided by the "caret" package in R.

5.4 Support Vector Machine

Given some labeled data for training, the SVM algorithm outputs an optimal hyperplane which classifies the new data. This hyperplane gives the largest minimum distance to the training examples. So it optimizes the margin between points at the edge and the hyperplane. This algorithm is applied to our dataset by using the "ksvm" function that is provided by the "kernlab" package in R.

5.5 NaïveBayes

Naïve Bayes is a simple technique for constructing classifiers. It is not a single algorithm for the classifier training, it is a family of algorithms based on a common assumption that the value of one feature is independent from the other features, given the class label. The advantage of this technique is that it only requires only a small number of training data to estimate the parameters needed for classification. This algorithm is applied to our dataset by using the "naiveBayes" function that is provided by the "e1071" package in R.

6. Experimental Results & Analysis

The algorithm for each classifier was applied to the training dataset to build a corresponding classifier. This classifier was then applied to the testing data to determine its accuracy. The accuracies obtained by each classifier was tabulated. The table of log (Table 1) is given in the Appendix.

As seen in the table, the best accuracy (98.98%) was provided by k-Nearest Neighbor Classifier. This classifier was then applied on the test data provided by the Kaggle "Leaf Classification" Competition. The Logarithmic loss obtained was 0.249.

7. Conclusion

We began by preprocessing the dataset. This included removing lesser significant columns and scaling the remaining data present in the data set. On doing so we observed that we obtain better results compared to just changing parameter values for various classifiers without any preprocessing.

We then applied five classifiers, namely, Bagging (Bootstrap Aggregation), Random Forest, k-Nearest Neighbor, Support Vector Machine and Naïve Bayes. Each

classifier was run multiple times on the preprocessed data set to obtain different accuracies. As seen under the Analysis section, k-Nearest Neighbor classifier provided the best accuracy of "98.98%". This classifier was further used in the Kaggle Competition to obtain a Logarithmic loss of "0.249".

8. References

[1]http://www.actapress.com/PaperInfo.aspx?paperId=455022

[2]http://stats.stackexchange.com/questions/18891/bagging-boosting-and-stacking-in-machine-learning

[3]http://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

[4]http://www.cs.cmu.edu/~epxing/Class/10701-10s/Lecture/lecture7.pdf

[5]https://www.techopedia.com/definition/5967/artificial-neural-network-ann

[6]http://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/

[7]https://www.r-bloggers.com/my-intro-to-multiple-classification-with-random-forests-conditional-inference-trees-and-linear-discriminant-analysis/

[8]http://stats.stackexchange.com/questions/15728/how-can-i-implement-multiclass-k-nn

[9]http://stats.stackexchange.com/questions/15728/how-can-i-implement-multiclass-k-nn

[10]http://machinomics.blogspot.com/2012/03/multiclass-svm-with-e1071_20.html

[11]https://mlr-org.github.io/mlr-tutorial/devel/html/multilabel/index.html

[12]http://scikit-learn.org/stable/modules/ensemble.html

[13]https://www.r-bloggers.com/a-brief-tour-of-the-trees-and-forests/

[14]https://www.r-bloggers.com/learning-kernels-svm/

[15]https://machinelearningmastery.com/

[16]https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

[17]https://www.kaggle.com/c/leaf-classification

APPENDIX

Table 1. Log of accuracies on applying various classifiers

Sampling#	Train/Test Percent split	Bagging Accuracy (%)	Random Forest Accuracy (%)	KNN Accuracy (%)	SVM Accuracy (%)	Naïve Bayes Accuracy (%)
1	80/20	80.30	95.95	**98.98**	96.46	**98.48**
2	80/20	87.37	**96.30**	96.46	95.55	88.38
3	80/20	84.84	89.75	98.48	97.97	90.40
4	80/20	**88.88**	93.55	97.47	**98.30**	85.85
5	80/20	85.67	92.87	95.95	97.47	88.88