Capstone Project Proposal

Anthony Le

February 4, 2017

Abstract

Featuring a Kaggle Competition, tackling leaf classification has many applications. With over a half million species of plants in the world,c lassification of plants have been a hard task which can lead to duplicate indentifications. This proposal demonstrates the feasibility and dataset of classifying the "random forest for the leaves".

1 Introduction

Featuring a Kaggle Competition, tackling leaf classification has many applications. With over a half million species of plants in the world, classification of plants have been a hard task which can lead to duplicate indentifications. Being able to classify leaves in this method will help problems such as determining whether a leaf is part a new species or whether the leaf is part of this genus or another genus. The idea of classification of this sort is based on leaf images and other extracted features such as shape, margin, and texture. Many of these features are not limited to leaves but can be similary extracted from images of faces, street signs, or numbers. There are various methods to extracting these features and classifying the set. A paper from MIT "Face Recognition with Support Vector Machines: Global versus Component-based Approach" by Heisele has successfully used Support Vector Machines to classify faces. Other similar work, such as from TensorFlow, shows deep neural networks classifying numbers based on its image. A leading education website, Udacity, teaches students how to use these same neural networks to classify street signs.

It should be clear how and why this problem should be solved. Also, some personal motivation includes understanding these machine learning techniques and how different methods might have different levels of variablilty.

1.1 Problem

The objective of this playground competition is to use binary leaf images and extracted features, including shape, margin and texture, to accurately identify 99 species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species. They also provide a fun introduction to applying techniques that involve image-based features.

The Kaggle competition mentions as a first step to "try building a classifier that uses the provided pre-extracted features. Next, try creating a set of your own features. Finally, examine the errors you're making and see what you can do to improve." In the end, classification will end up using the provided images since score of Kaggle competitions rely solely on those images and creating features from the images and at the same time using the features provided.

1.2 Dataset

The dataset can be found in the kaggle database section of the competition (link below). There are 1583 images of leaf specimens or 16 samples of 99 different species. Some preprocessing have been made to convert images to grey-scale against a white background. Three features are provided per images - "a shape contiguous descriptor, an interior texture histogram, and a ne-scale margin histogram" https://www.kaggle.com/c/leaf-classification/data

1.3 Solution

The solution can be found using supervised learning methods. With scikit-learn, various methodologies such as KNN, Decision Trees, Supper Vector Machines, and much more can be used for classification of leafs. However, deep learning may be used in conjunction with convulational nueral networks to classify leafs. First, preprocessing must be done. This will include changing the leaf images to more easily identifiable and classifiable images. This could be using edge-detection to determine the general structure of the leaf or by looking at some other feature from the leaves. Also, we will want to separate the sets into a test and training set in order for use to determine the accuracy of our training. After preprocessing is done, this will create the set of features we will want to use with our classifiers. The classifiers range from KNN, SVM, Decision Trees, Deep Convultion Networks, to much more. The classifiers mentioned are likely to give the best results. Tensorflow and Sci-kitlearn will be the essential toolboxes needed to preprocess the data, fit the data into the classifier, and ultimately classifying the data accordingly.

1.4 Benchmark Model

The benchmark model for this problem is scoring below a 30 in the Kaggle Competition. The Kaggle Competition uses a multi-class logarithmic loss to estimate the score. As a note, the score closest to 0 is the best scoring. Another benchmark for this problem is using an SVM to classify my results and continuing to see if any other classifiers have a similar score. As a note, due to the number of images, deep learning may not prove as beneficial as SVM since deep learning will likely need more images to correctly identify images.

1.5 Evaluation Metrics

Evaluation can be directly made via Kaggle by uploading the kernel or results to the website. The evaluation is based on the log loss as stated in the benchmark model.

$$logloss = -(1/N) \sum_{i=1}^{N} \sum_{i=1}^{M} y_{ij} log(p_{ij}) \tag{1}$$

"where N is the number of images in the test set, M is the number of species labels, *log* is the natural logarithm, y_{ij} is 1 if observation i is in class j and 0 otherwise, and p_{ij} is the predicted probability that observation ii belongs to class j." Various other evaluation metrics can be used such determining accuracy between the training and test sets and comparing accuracies of different classifiers within one another.

1.6 Project Design

Project design will follow methodology from the supervised learning class offered by Udacity. This class uses the scikit-learn library and follows a format of preprocessing data, fitting data, and ultimately classifying data for our use.
My workflow will be as follows:

1. Collect dataset from kaggle.

2.Process the dataset in ways previously mentioned.

3. Divide the set into train, validation and test set.

4. Implement the SVM classifier to that dataset.

5. Obtain the results with benchmark by submitting to the kaggle leaderboard and by noting down the accuracy.

6. Chose a different classifier and repeat step 5.

7. Compare results with the other classifiers.

8. Submit the final solution to the Kaggle's leaderboard and to Udacity.

After being able to test the SVM classifier to the dataset and understanding how to submit to the kaggle leaderboard, implementing and determining the best classifiers should be relatively simple.