Enhancing the Accuracy of the Random Forest Algorithm

SISC Machine Learning Group 6

Introduction

The random forest algorithm is an adaptation of the decision tree algorithm, except with bagging implemented. Multiple subsets of the complete dataset are used to produce classifications, with the average then taken to produce a more accurate result.

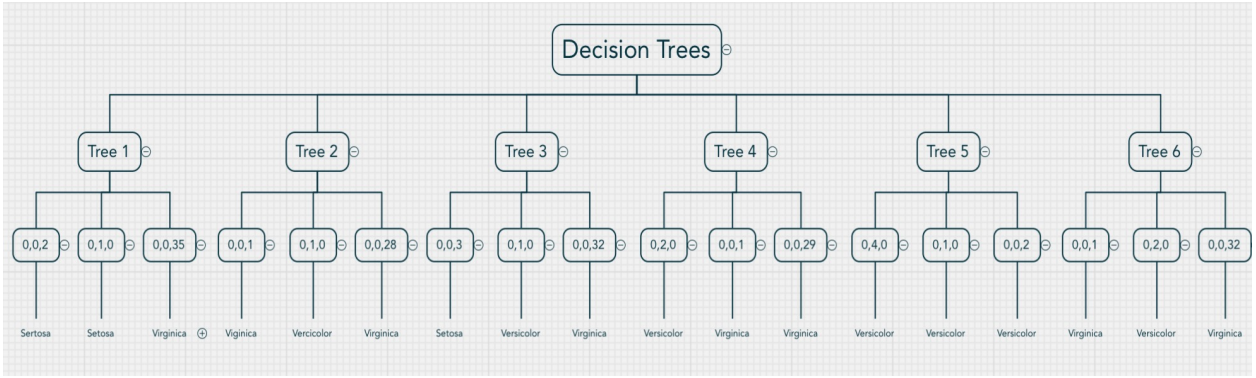
Our demonstrations are performed on the well-known iris dataset.

Methodology

1. Scikit-learn's implementation of the decision tree algorithm is built upon, turning it into a random forest algorithm.
2. In each iteration, the complete dataset is split 70%/30%, with the 70% further split into smaller, perhaps overlapping subsets to train the algorithm, and the 30% used to test the algorithm.
3. The number of iterations, as well as the size of the subsets, and number of subsets generated, is varied.

Results

1. The generation of a forest (6 trees), as opposed to a single tree, is found to have significantly increased the accuracy of the algorithm.



Additional Concepts

- The inclusion of a larger dataset can make classifications more refined and accurate.
    - This will lessen overlap of individual data sets (for each tree in the forest), and provide sufficient data to overcome the deviation caused by outliers.
    - The dataset should be refined (manually) to rule out datapoints that are unnecessary or unimportant.
    - This, however, is done at the expense of the efficiency of the algorithm, as more data points have to be sorted.
- Pruning unnecessary classifiers can simplify the algorithm.
    - Too many classifiers can lead to the over-fitting of each tree. Despite this being overcome by the taking of an average of all individual classifiers, this effect may only be superficial.

Enhancing the Efficiency of the Random Forest Algorithm

SISC Machine Learning Group 6

Methodology

1. The runtime of the code is measured before and after the improvement (to vary the number of iterations, the size of the subsets, and the number of subsets generated) is made.

Results

1. The generation of a forest (6 trees), as opposed to a single tree, is found to have the smallest increase in the runtime of the algorithm. This is while maintaining the highest accuracy.

Additional Concepts

- Feeding less data points will reduce the runtime.
    - This, however, may compromise the accuracy of the final classifier.
    - That said, the accuracy of the classifier increases at ln(x) rate, so the trade-off between runtime and accuracy eventually is viable.
    - In addition, due to the classifier algorithm, each new set will take a different learning path and therefore the time taken to process will differ

- Making sure there are no outliers in the data to make the outcome less stressful to decipher.
    - This is difficult as all data will be different
    - The larger the sample size fed, the more accurate a prediction the programme can make
    - However, it cannot be too large as it will take up more processing power for little gain

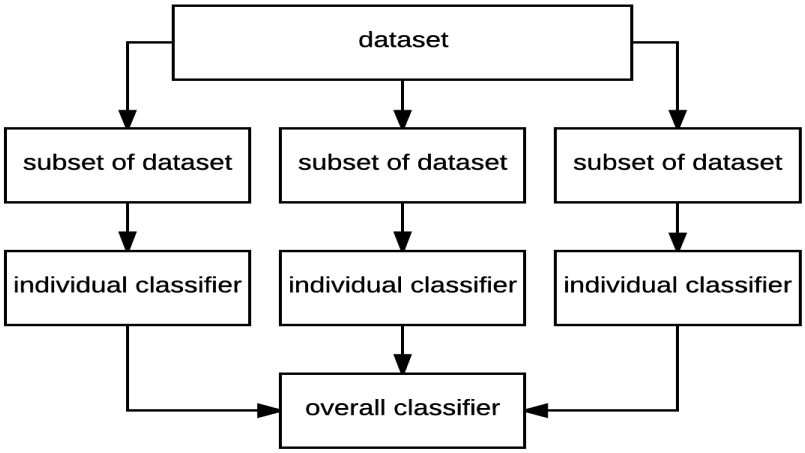
- Ensuring that there are enough trees in the “Forest”
    - Allowing for a representative result that is less likely to be biased towards a certain result.
    - Too many and replicates may appear, resulting in bais.
    - Adequate amount, such as 6, to allow for enough representation of all data, yet avoiding a result that favours the classification of a certain category

Enhancing the Scrutability of the Random Forest Algorithm

SISC Machine Learning Group 6

Methodology

1. Multiple comments are added into the final python code. This explains the operation of the code in a step-by-step manner.
2. Visualisation of the code via diagrams aids one in understanding the piece of code.



Results

1. See code below
2. High throughput allows for a typical accuracy of 96%, and at 70%/30% split ⅘ of the sets are often "voted" for and are identical in classifier.
3. Typical time of processing of 6-Tree "Forest" is timed to be around $1/10^{th}$ of a millisecond, which is not significantly longer than running a singular tree, which is around the same processing time.

Python / Scikit Code

SISC Machine Learning Group 6

Code for Random Tree Algorithm (by scikit)

1.11.2.2. Extremely Randomized Trees

In extremely randomized trees (see `ExtraTreesClassifier` and `ExtraTreesRegressor` classes), randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias:

```
>>> from sklearn.model_selection import cross_val_score
>>> from sklearn.datasets import make_blobs
>>> from sklearn.ensemble import RandomForestClassifier
>>> from sklearn.ensemble import ExtraTreesClassifier
>>> from sklearn.tree import DecisionTreeClassifier

>>> X, y = make_blobs(n_samples=10000, n_features=10, centers=100,
...     random_state=0)

>>> clf = DecisionTreeClassifier(max_depth=None, min_samples_split=2,
...     random_state=0)
>>> scores = cross_val_score(clf, X, y)
>>> scores.mean()
0.97...
```

Code for Random Forest Algorithm

Random Forest.py - /Users/DarellCHUA/Documents/Random Forest.py (3.6.1)

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
Y = iris.target

from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2)

my_classifier = tree.DecisionTreeClassifier()
my_classifier.fit(X_train, y_train)

predictions = my_classifier.predict(X_test)

from sklearn.externals.six import StringIO
dot_data = StringIO()
tree.export_graphviz(my_classifier,
                         out_file=dot_data,
                         feature_names=iris.feature_names,
                         class_names=iris.target_names,
                         filled=True, rounded=True,
                         impurity=False)
print("number ")

from sklearn.cross_validation import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2)

my_classifier0 = tree.DecisionTreeClassifier()
my_classifier0.fit(X_train, y_train)

predictions = my_classifier0.predict(X_test)

from sklearn.externals.six import StringIO
dot_data0 = StringIO()
tree.export_graphviz(my_classifier0,
                         out_file=dot_data,
                         feature_names=iris.feature_names,
                         class_names=iris.target_names,
                         filled=True, rounded=True,
                         impurity=False)
print("number 0")
```

Ln: 1 Col: 23