An exploration in Decision Trees, Random Forests and AdaBoost

Carl Aiau

Introduction

In this assignment, I explored the learning capabilities and performance of three popular and commonly used machine learning (ML) models: Decision Trees, Random Forests and AdaBoost. Random Forests and AdaBoost both extend the original Decision tree model. Rather than using the prebuilt Sklearn algorithms for these ML models, I wrote my own from scratch using Python and Numpy. All three ML models were assessed against four classification datasets from the Sklearn library. Three of these data sets were non-binary classification tasks, which AdaBoost cannot learn (without further ensemble methods), but I restricted the data of these sets to binary classification whenever one of the models assessed was AdaBoost. I spent most my experimental time adjusting the parameters of individual models.

Methods

**Decision Trees**

A Decision tree is a binary tree, where every node on that tree is either a decision, which branch’s the sample set, or a leaf which has no further splitting. We begin at the root node of the tree, and this node will ask a question about the features of the input data. Depending on the answer we will follow the true or false branch down one more layer. This is a recursive method that continues until you reach a leaf node, or reach a stopping criterion. The beauty of decision trees is they are interpretable. You train a decision tree at every node by determining which question (feature, and value of that feature) can be asked to result in the biggest information gain from splitting the data. When predicting a classification, you follow the branches of the trained tree choosing true and false branches depending on the test data values until you reach a leaf node.

My decision tree is greedily trained, it iterates through every feature of the input data, and then iterates over every value of that feature, and attempts to partition the data on every iteration. The algorithm will then choose the partition that resulted in the greatest gain of information at every decision node. By default, a decision tree will keep splitting the data until every node results in containing only one class of data, but this is easily prone to overfitting by splitting on noise and outliers. This overfitting can be greatly reduced and the generalisation potential of the tree increased by implementing a stopping criterion. This stops the tree from partitioning once the number of input data to a node becomes less than some criteria. I choose to use a stopping criterion of 10% of the total input data samples. When exploring the decision tree algorithm via my CLI, I made this value adjustable. For Random Forests and AdaBoost, there is no stopping criterion as both methods should prevent the contribution of individual trees overfitting at the meta-learning (ensemble) level.

**Random Forests**

A Random forest is an ensemble method, a forest is a collection of decision trees. Each decision tree is trained on a random subset of the features, and a random subset of the values of those features. After training, the final prediction of a random forest is an aggregation of all the contained trees. Because we’re aggregating, we remove the stopping criterion of each decision tree, we allow them to branch deeper. This is because each decision tree within the ensemble is given a random set of features and values, so in combination the overfitting of the decision trees individually isn’t a “global overfitting”.

By default, the subset of features chosen is the square root of the original features, and I am training the trees with a 50% randomised subset of the data. By default, the number of trees in the forest is 10. For the comparison methods, these are the parameters used. For the individual exploration of Random forests both the size of the forest and the feature subset count are adjustable.

**AdaBoost**

AdaBoost is short for adaptive boosting. AdaBoost is used in conjunction with other types of learning algorithms to improve performance. For my implementation, I choose to use AdaBoost with Decision stumps, which is a 1 layer deep decision tree. If the chosen weak learner has stronger than 50% predictability, the overall model can be proven to converge to a strong learner. Similarly, to a Random Forest, the end prediction is based on the combined output of an ensemble of learners. The difference is that the AdaBoost sum is weighted, based on the training error that each individual learner encountered while learning. Therefore, when doing the weighted sum, the learners, that had high training error will have less weight on the AdaBoost prediction than the learners that has less error. The Boosting refers to how the algorithm learns (magic). The sample choice for each iteration is randomly drawn. After each iteration, the learnt decision stump is used to predict the classification of the training data, and incorrect classifications are identified. These incorrect classifications create a training weight, which emphasizes the further training of these incorrect classifications. For each iteration, the sample choice is drawn randomly, from this weight distribution. For my implementation, I choose 100 decision stumps, and this is the value used for model comparisons, but it is tuneable for individual model exploration.

**Evaluation**

I evaluated the performance of each model via SKlearns cross validation, taking a 80/20 train/test split:

cv = ShuffleSplit(n\_splits=100, test\_size=0.2)

scores = cross\_val\_score(classifier, x, y, cv=cv)

For the comparisons between the three models, I restricted the datasets to the data that only was associated with class 0 or 1. For the comparison between Decision Trees and Random Forests I used the full non-binary datasets. For both binary and non-binary comparison testing I ran the process for 100 samples. When doing individual model exploration (parameter tuning), decision trees and random forest both use the full datasets. Whereas AdaBoost has uses the restricted binary datasets.

**Results**

Binary Comparison Between the three models, 100 iterations.

|  |  |  |  |
| --- | --- | --- | --- |
| **Iris** | **Performance** | **Cancer** | **Performance** |
| Decision Tree | 1.00 (+/- 0.00) | Decision Tree | 0.90 (+/- 0.04) |
| Random Forest | 0.99 (+/- 0.05) | Random Forest | 0.95 (+/- 0.04) |
| AdaBoost | 0.99 (+/- 0.04) | AdaBoost | 0.95 (+/- 0.04) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Wine** | **Performance** | **Digits** | **Performance** |
| Decision Tree | 0.93 (+/- 0.10) | Decision Tree | 0.98 (+/- 0.02) |
| Random Forest | 0.96 (+/- 0.09) | Random Forest | 1.00 (+/- 0.01) |
| AdaBoost | 0.97 (+/- 0.04) | AdaBoost | 1.00 (+/- 0.01) |

Non-Binary Comparison between Decision Trees and Random Forests, 100 iterations

|  |  |  |  |
| --- | --- | --- | --- |
| **Model / Dataset** | **Iris** | **Wine** | **Digits** |
| **Decision Tree** | 0.94 (+/- 0.07) | 0.84 (+/- 0.13) | 0.55 (+/- 0.07) |
| **Random Forest** | 0.93 (+/- 0.09) | 0.94 (+/- 0.08) | 0.82 (+/- 0.06) |

Here on the Wine and Digit datasets we are beginning to see some substantial increases in performance between an individual decision tree and the random forest ensemble.

For further exploration of individual models, I choose to explore AdaBoost on Cancer as it is the dataset that had the worst AdaBoost performance. I ran each cross validation 50 times so the results are more prone to variance than I would have liked but this was due to time (and computational) restrictions. Initially I tested to see how AdaBoost would perform with a limited number of stumps, then also tested how AdaBoost would perform with many stumps.

I was surprised at how steep the accuracy increased between 5 – 25 stumps, but was quite underwhelmed by the performance increases after this. Above 50 stumps, the test performance only slightly increases by around 1% to 100 stumps, and then above that I’m not confident that the model performed better than variance. The Standard deviation graph was interesting, you can see the training variance continue to decrease, but test remains constant after 50 stumps. It’s as though the model has reached its peak performance from tuning the number of stumps, and for further performance increase we would need to start adjusting other parameters such as error criterion of the trees, or sub setting the samples for each stump.

I took a closer look at Random Forest on the non-binary digit classification task. RF only achieved 82% classification when using the default number of trees in the ensemble. As you can see this increased steadily as the number of trees in the ensemble was increased, but caps at around 70. The inefficiency of my implementation made exploring above 100 with any reasonable sample size painfully slow. This data was generated for 50 iterations at each ensemble size.

After tinkering with the number of trees in the ensemble, I then played with the number of subset features, I used an ensemble count of 20 as it gave us quite a big boost in performance from default without a huge cost in computation. The performance jumped substantially from 8 to 16, but after that I don’t think the increased learning time is worth the burning silicon. The feature performance data is only based on a sample size of 25 iterations so is prone to more variance than the ensemble adjustment data.

I think a good performance / computation trade-off is made with a Random Forest with 20 ensembles and 16 features, when compared to the base line RF at 83% test accuracy, 95% is a substantial increase. Each iteration at these parameters takes around 24 seconds to run on my 2012 MacBook, as opposed to 64 features which was taking upwards of 300 seconds.

**Discussion / Conclusion**

Both Random Forests and AdaBoost showed increases in performance over the stand-alone decision tree. Because of how AdaBoost was trained, it was not extended to multi class classification, so the results from our three model comparisons had to be restricted to binary subsets of the datasets which made the results underwhelming. Cancer was the only example where AdaBoost could be seen to be clearly beating the stand-alone decision tree. The differences between Random Forest and Decision Trees is more evidently shown when they are tested on non-binary datasets, specifically Digits, with 82% (95% after hyper parameter tuning) versus 54% respectively. For further exploration, I would like to explore using AdaBoost in a multi class problem via multiple binary AdaBoost classifiers and assessing the ensembles overall performance against Random Forest on non-binary data. I have learnt a lot from building these machine learning algorithms from scratch and AdaBoost is quite intuitive now. I would also like to extend my CLI to encompass the tuning of the individual trees within Random Forests and AdaBoost, but will replace my inefficient python with SKlearn methods.

Review