Implementation and Analysis of Random Forests

Jae Lee
School of Computing Science
Simon Fraser University
Burnaby BC V5A 1S6

Richard Mar
School of Computing Science
Simon Fraser University
Burnaby BC V5A 1S6

Robin White
School of Mechatronic Systems Engineering
Simon Fraser University
Surrey BC V3T 0A3

1 Introduction

In machine learning, there is often a tug-of-war between bias and variance; having high accuracy to observed data but not to lose generalization (or over-fit) to unseen data. This is often referred to as the "bias-variance tradeoff" and its consideration is a significant part of properly engineering machine learning algorithms. Often, regularization is used during training of a machine learning algorithm where there is an addition to the loss function to represent a cost to complexity and force generalization, or in the way of neural networks, random dropout of neurons to force generalization [1]. Regularization and similar methods add to the complexity of the algorithm and can lead to loss in accuracy of predicting on the training data.
In the early 90's, Tin Kam Ho from Bell Labs published a series of papers where he showed that by combining independent learners in a unique way, the accuracy of classifying handwritten digits monotonically could be increased without suffering from over-adaptation to the training data. [2, 3, 4] The application of this method to decision trees in Ho's '93 paper marked the introduction of random forests to the community. [2] Decision trees are simple yet effective classifiers, with high execution speed and the resulting tree can be easily understood by a human; however, they are subject to over fitting and lose generalization to unseen data. Some methods such as pruning have been used previously to try and increase generalization; however, methods such as these usually come with a loss in accuracy toward training data. By using principles of stochastic modeling, Ho showed that tree-based classifiers could be arbitrarily expanded for increases in accuracy on unseen testing data without loss in training data accuracy. A characteristic which is still unique among many machine learning classifiers. The concept is that multiple unique learners can compensate for the bias of a single learner and so trees are constructed by randomly selecting subspaces of the feature space. In this way, each tree generalizes in a different way.
Random forests have been applied to a variety of machine learning tasks including classifications in ecology and geosciences, image segmentation in medical applications, business analytics, sporting analytics, as well as the unmentioned number of general data science applications. [5, 6, 7, 8, 9]. These traits of high accuracy and resistance to over fitting that random forests possess are fascinating and so we would like to further our knowledge in how they are able to achieve these feats.

2 Approach

In order to develop our knowledge of random forests and investigate their learning capabilities, we implement the random forest algorithm in python and apply it on a specific dataset. Our dataset of choice is the hockey player dataset from class as it possesses discrete and continuous variables which allow us to use the random forest algorithm as a classifier and regressor on the same dataset. In order to determine whether or not our implementation is correct, we compare it against other freely available implementations. Once we are confident that our implementation is correct in applying ensemble learning, we explore varying specific learning parameters for random forests to understand their impact on random forest's ability to learn from a training set and predict on a test set. Finally, armed with knowledge on how random forest's learning parameters should be set for optimal results, we compare random forest's ability to predict attributes from various hockey players to see how it fairs relative to other well-known machine learning algorithms.

3 Experiments

3.1 Pre-processing of the Data

Prior to carrying out our experiments, specific columns in the hockey dataset were dropped and categorical attributes were converted into dummy variables as described in Assignments 1 and 2.

3.2 Implementation Comparisons

Table 1: Comparison of different random forest classifier implementation's ability to predict GP_greater_than_0.

Machine Learning Package	Accuracy	Train Time
Weka	69.4%	<1sec
scikit-learn	65.9%	<1sec
our implementation	68.6%	14sec

Table 2: Parameters used to generate the results for our implementation of random forest in Table 1.

Parameter	Value
Trees	32
Depth	5
Features	5
Split Size	2
Splitting Criteria	Gini Impurity

The accuracy of our implementation of random forest against Weka's and scikit-learn's is compared to demonstrate its correctness. Settings were chosen to be as similar as possible to other implementations when making the comparisons shown in Table 1. The Weka implementation used bagging of J48 decision trees to best correlate to the implementation used in this project. The Weka random forest implementation uses other additions such as choosing uncorrelated trees to optimize the results. The Weka platform does not give an option for maximum tree depth; however, to better compare to this project's implementation, binary splitting was used.

From Tables 1 and 3, it can be seen that all three implementations of random forest yield similar results using similar parameter values (as is possible by the user). The parameters for our implementation were as seen in Table 2.

It was noted however that when performing on standardized data, such that values are centered by their means and divided by the standard deviation, the accuracy was significantly lower for all three implementations, around 55% accuracy (not shown here). This can be due to the sensitivity of the splitting using Gini impurity or entropy with small values, or perhaps some information is lost on the impact of certain features. The time taken for training is difficult to compare since coding implementations are different (python vs C), however to improve speed, multi-processing was implemented. Using 4 processes running the same parameters as shown above for target class GP_greater_than_0, the time of $\approx$ 14 seconds is slower, however not unreasonable, this addition was a great improvement over a single process taking 41 seconds.

Table 3: Comparison of Random forest regressor for sum_7yr_GP

Machine Learning Package	Mean square error	Train Time
Weka	9185.3	<1sec
scikit-learn	9239.5	<1sec
our implementation	9446.5	1min 11sec

As a note, for classification we used the J48 decision tree and bagging method from Weka, however this tree type is not able to be used for regression. As such the REPTree was used instead. Our implementation of regression splitting uses mean squared error as a scoring function.[10]

3.3 Parameters Exploration

3.3.1 Number of Trees

Random forests utilize an ensemble method where a collection of weak learners is combined to produce a more accurate and robust model. We explore this by investigating the classification accuracy of our random forests implementation by varying the number of decision trees which are combined to produce the final prediction of our model. We keep parameters such as maximum tree depth, number of features, and minimum split size constant, with values 5, 5, 2 respectively.

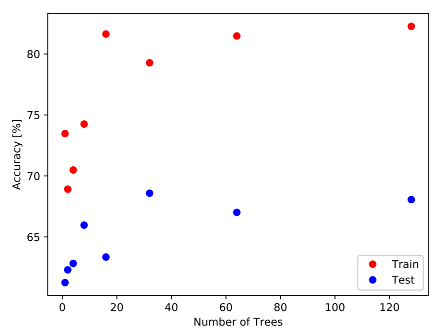

Figure 1: Showing random forest inhibition to produce over fitting errors even with increased complexity by large number of trees.

The ability of random forests to resist over fitting is largely due to the number of trees and features, which is demonstrated in Figure 1. Randomization increases bias but makes it possible to reduce the variance of the corresponding ensemble model through averaging. [11]

3.3.2 Tree Depth

The depth of each decision tree used to make up the ensemble used in random forests has a large impact on the accuracy of the model. We investigate this by comparing the accuracy of our model implementation as a function of the maximum depth of each tree. That is, the tree can have a depth that is less than this value, depending on the features used to build the tree, however once it reaches this maximum depth, the leaf is forced to be a terminal and predict the target class. We keep other parameters such as number of trees, number of features, and minimum split size constant, with values 64, 5, 2 respectively.

It has been shown that by increasing the maximum depth of each tree, a level of over fitting can be present, and similarly, having shallow trees can produce low-confidence predictions. [12] It is thus important that an appropriate value of the maximum depth be used, as its optimal value is related to the problem complexity. We notice a small decrease in the accuracy at a max depth of 20, as

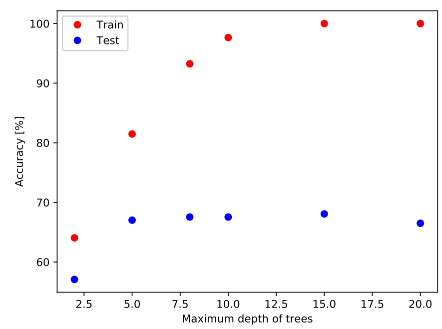

Figure 2: Showing accuracy of random forest with 64 trees with varying maximum depth of each decision tree. It should be noted that there is a slight decrease in the test accuracy at a max depth of 20 which can possibly be related to over fitting on the training data which reaches 100% after 15.

shown in Figure 2, which suggests that random forests are still susceptible to over fitting, albeit less sensitive than many other machine learning algorithms without active regularization.

3.3.3 Features per Tree

It is discussed in literature that an optimal value for the number of features to be used is $\frac{\ln(M)}{\ln(2)}$ where M is the total number of features. [13] To investigate this we compare the classification accuracy while varying the number of features, keeping all other parameters such as number of trees, maximum depth, and minimum split size constant, with values 64, 5, 2 respectively.

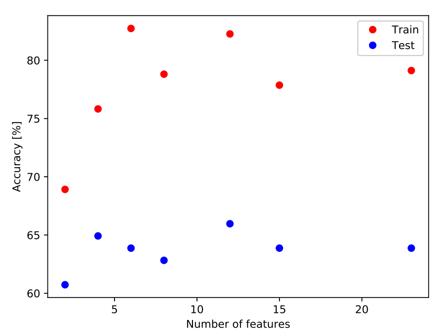

Figure 3: Showing the impact of the number of features used when constructing decision trees in random forest with 64 trees and max depth 5.

Following our results, shown in Figure 3, we observe a maximum train accuracy for 5 features, which agrees with literature for an optimal number of features as M=23 for our pre-processed dataset. We observe, however, that the number of features used has less of an impact on the total accuracy when comparing to the above observations for varying number of trees and maximum tree depth. The maximum depth for optimal test accuracy also does not agree with the training value, and the variance between each run makes drawing a conclusion on the over-all impact of this parameter difficult.

3.3.4 Grid Search Parameter Optimization

Following the observations after varying each parameter individually, a grid search method was employed to find the optimal values for number of trees, maximum depth, and number of features for our implementation of random forest. The range of values used for the grid search were as seen in Table 4.

Table 4: Range of values used for the grid search of our implementation of random forest.

Parameter	**Values**
Trees	2,8,16,32,64,128
Depth	2,5,8,10,15,20
Features	2,5,8,10,15,23

With optimal values shown in Table 5. Figure 4 shows the effect of the different parameters on the accuracy of the classifier for the test set. As can be observed, there are some local regions of high accuracy. This is particularly true when looking at the areas where number of features is close to $\frac{\ln(M)}{\ln(2)}$ which is approximately 5 in our case. It is surprising that the optimal number of features is very low with a value of 2. This may be a result of creating a higher chance of uncorrelated trees which has been shown in literature to improve accuracy. [14]

Table 5: Optimal values found for our random forest implementation by grid search

Target class	**Max depth**	**Num features**	**Num of trees**	**Accuracy measure**
GP_greater_than_0	15	2	64	71.2%
sum_7yr_GP	5	13	10	9946

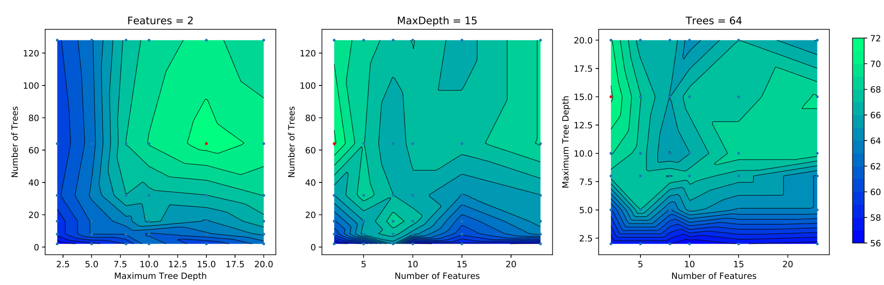

Figure 4: Showing results from grid search with effect on test set accuracy of our random forest implementation when predicting GP_greater_than_0

Comparing our implementation of random forest to that of scikit-learn implementation, we further compare against the optimal values found by grid search, shown in Tables 5 and 6. As can be seen, the values are quite different, although result in similar accuracy. A high number of trees is consistent between both showing the over-all effect of the random forest algorithm on improving performance for the classifier implementation. The difference between the values can be due to the different algorithms used and shows that finding optimal values for the specific algorithm is an important consideration when running any machine learning algorithm implementation. Using optimal values for learning, time for 4 processes of 20seconds is a significant improvement over a single process taking 1min 4sec for training GP_greater_than_0, and comparing to Table 1 the effect of increasing the number trees can be seen in the difference in time. After training however, prediction time is considerably lower, taking ≈1sec regardless. Similarly, training a regressor implementation on sum_7yr_GP target label, similarly observations are made. The performance of an ensemble learner is dependent on the accuracy of each component learner and the diversity of the components, especially when using a small set of trees which may be limited due to computational cost. The randomization can cause occurrence of bad predicting trees as well as correlated trees which can lead to poor ensemble decisions, which can be observed when performing multiple training runs using the

Table 6: Optimal values found for sci-kit learn's random forest implementation by grid search

Target class	Max depth	Num features	Num of trees	Accuracy measure
GP_greater_than_0	8	5	128	67.5%
sum_7yr_GP	8	5	64	9270.5

same parameters that can lead to different accuracy results. Attempts have been made to improve the performance of this model by building a forest of only uncorrelated high performing trees. [14]

3.3.5 Splitting criteria

When building a decision tree, the splitting criteria by which each level is formed to yield the greatest information gain can be calculated by two common methods; Entropy and Gini impurity. It has been discussed in literature that there exists a minimal difference between the effect of accuracy, however Gini impurity function is computationally less expensive than Entropy since Entropy requires additionally calculating the logarithm of percentage of each class labels in the child nodes. The difference between the splitting criteria is investigated through the implementation of both entropy and Gini impurity in our implementation of Random forest, with results shown in Table 7. Both models were run on a single process.

Table 7: Comparison of Entropy and Gini impurity for splitting criteria

Number of trees	Gini impurity	Entropy
4	1min 23secs	1min 47secs
16	3min 58secs	4min 26secs
32	6min 19secs	7min 23secs

As shown in Table 7, using the Gini index as the score function is indeed noticeably faster than Entropy.

3.4 Machine Learning Algorithm Comparisons

For comparison, a decision tree classifier and a support vector machine (SVM) were trained and tested on the data. The test set accuracy of the decision tree was 64% while the test accuracy of the SVM was 68%. And it is clear that our random forest implementation performed better than both. Linear regression was also compared to the random forest regressor. This used interaction terms in the pre-processing of the data and resulted in a mean square error of 11186.6. This also uses regularization in the algorithm implementation. This value is in the range of the prediction from random forest, however a significant improvement is made using a random forest implementation.

4 Conclusion

Through this project, we have explored and discovered many characteristics of random forests ability to learn. We compared our algorithm to two common machine learning libraries, Weka and Scikit-Learn, and demonstrated that our algorithm is able to predict with similar accuracy and that the implementation captures the principles behind random forests properly. Next we were able to explore the effect of parameters individually, by holding others constant. For simplicity we only considered classification. We observed that there was an improvement in prediction accuracy with an increased number of trees, and that no over fitting was noticed as the number was increased, which is unlike many other machine learning algorithms. Similarly, as we increased the number of features used to learn per tree, no over fitting was apparent. However, increasing the maximum depth of the trees in the forest revealed that mild over fitting does occur at a high depth, which is consistent with what is found in literature. After learning the effect of adjusting three of the learning parameters individually, a grid search using these same three parameters to find optimal values for high prediction accuracy was performed. Surprisingly, the number of features for maximum classification accuracy was only 2. This may be a result of creating uncorrelated trees where there is a low chance of having similar features when only 2 features are used per tree. Finally, we compared the random forest algorithm to other machine learning algorithms. For a single decision tree the accuracy was indeed lower as well as for SVM, which was most likely attributed to over fitting. From our experiments we observed several areas where improvements can be made in future work. The running time can be decreased by optimizing split function subroutines and modifying the data structures to use Cython or C. In addition, it would be interesting to implement algorithms found in literature designed to minimize the correlation between the trees in the random forest, which may improve the accuracy of our random forest implementation.

Contributions

All authors contributed equally.
See GitLab repository here for specific commits:
https://csil-git1.cs.surrey.sfu.ca/rkm3/mlclass-1777-randomforest

References

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[2] T. Ho, "Recognition of handwritten digits by combining independent learning vector quantizations," *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 818–821, 1993.

[3] T. Ho, "Random decision forests," *Proceedings of the Third International Conference on Document Analysis and Recognition*, pp. 278–282, 1995.

[4] T. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[5] D. R. Cutler, T. C. Edwards, K. H. Beard, A. Cutler, T. Kyle, J. Gibson, J. J. Lawler, H. Beard, and T. Hess, "Random Forests for Classification in Ecology," *Ecology*, vol. 88, no. 11, pp. 2783–2792, 2007.

[6] J. R. Harris and E. C. Grunsky, "Predictive lithological mapping of Canada's North using Random Forest classification applied to geophysical and geochemical data," *Computers and Geosciences*, vol. 80, pp. 9–25, 2015.

[7] G. Luo and K. Wang, "A combined random forest and active contour-model approach to fully automatic segmentation of the left atrium in volumetric MRI," *BioMed Research International*, vol. 2017, 2017.

[8] N. Ghatasheh, "Business Analytics using Random Forest Trees for Credit Risk Prediction: A Comparison Study," *International Journal of Advanced Science and Technology*, vol. 72, pp. 19–30, 2014.

[9] D. Lock and D. Nettleton, "Using random forests to estimate win probability before each play of an NFL game," *Journal of Quantitative Analysis in Sports*, vol. 10, no. 2, pp. 197–205, 2014.

[10] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*. Wadsworth and Brooks/Cole Advanced Books and Software, 1984.

[11] S. Formann-Roe, "Bias and variance," Jun 2012.

[12] A. Criminisi, J. Shotton, and E. Konukoglu, "Decision Forests for Classification , Regression , Density Estimation , Manifold Learning and Semi-Supervised Learning," *Microsoft Research technical report*, vol. 7, pp. 81–227, 2011.

[13] L. Brieman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[14] S. Bharathidason and J. C. Venkataeswaran, "Improving Classification Accuracy based on Random Forest Model with Uncorrelated High Performing Trees," *International Journal of Computer Applications*, vol. 101, no. 13, pp. 26–30, 2014.