Random Forest and SMO Learners

Ethan Robards, Joe Shymanski

February 19, 2022

Abstract

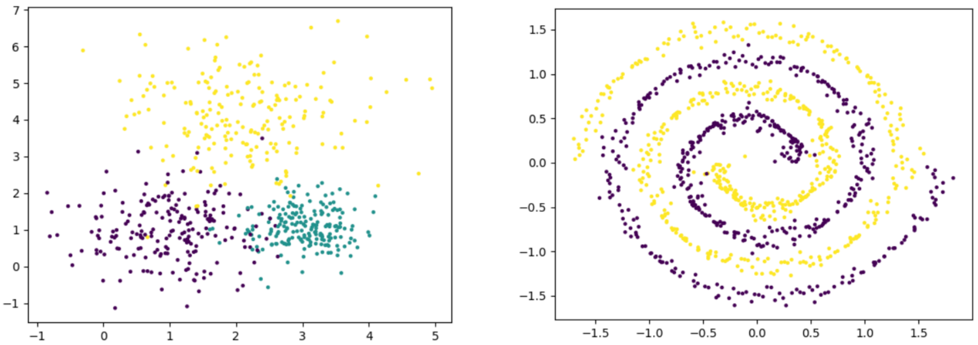
This paper discusses the implementation of the Random Forest and Sequential Minimal Optimization (SMO) algorithms to learn an ensemble of Decision Trees and Support Vector Machines (SVMs), respectively. The learners will then be used to classify six different datasets spanning over several different domains. The parameters of each model will be tuned and their performance will be optimized on each dataset.

1 Introduction

The Random Forest is an ensemble method that takes results from many Decision Trees and gives a result through a majority vote [1]. This reduces the variance that a single Decision Tree can provide by tempering it with many others, creating a powerful and useful predictive model in comparison. We evaluate the Random Forest classifier with several datasets, two artificially-generated and four real-world datasets.

2 Datasets

The Adults dataset contains generic demographic information for each person as well a class indicating whether or not they make over $50,000 a year. The Zoo data have a more physical feature set for classifying different species of animals. The Blobs and Spirals datasets contain points in two-dimensional space where the three blob classes and two spiral classes are fairly distinct with some overlap. These can be seen in Figure 1. The MNIST Digits



(a) Blobs

(b) Spirals

Figure 1: Plots for both auto-generated datasets.

dataset and the Letters dataset both seek to classify handwritten images of alphanumerics. The difference is in the number of features. The Digits data contain all 784 pixels as the features with intensity values ranging from zero to 255. The Letters dataset, however, contains only 16 generic summary features that range from zero to 15 in intensity. Table 1 contains the relevant counts for all the datasets.

Dataset	Continuous Features	Discrete Features	Classes	Examples
Adults	6	8	2	32561
Blobs	2	0	3	600
Digits	784	0	10	42000
Letters	16	0	26	20000
Spirals	2	0	2	1000
Zoo	0	16	7	101

Table 1: Count summaries for each dataset.

3 Models

3.1 Random Forest

The Random Forest learner can operate on virtually any domain, albeit with varying success. It is capable of both classification and regression. However, for the purposes of this paper, all datasets are used for classification.

Random Forest accepts a number of parameters in order to function optimally. The only parameter which is used exclusively in the Random Forest logic is the number of Trees to grow in the Forest. The rest are passed directly to the Decision Tree learner. The first few are the examples and features of the dataset to be classified along with which of three measurements of impurity to use: misclassification, Gini, and entropy. The last three are used to tune the learner. The minimum subset proportion dictates how small a randomly generated subset of a continuous feature's domain can be. Both maximum Tree depth and minimum number of leaf examples act as stopping criteria for the learner.

During each iteration of the Random Forest algorithm, a random Decision Tree is learned. To ensure its uniqueness, the learner chooses between a random subset of the features left, F, the number of which is determined by $\lfloor \log_2 |F| + 1 \rfloor$. If one of the features is continuous, then another random subset is taken on its domain according to the minimum subset proportion parameter. Stopping criteria are also used to prevent overfitting and guarantee more variation within the Forest. The rest of the Decision Tree learner is exactly the same as a generic learner meant to build a single optimal Tree.

3.2 SMO

We will explore the SMO learner in the coming weeks.

4 Experimental Results

The accuracy for both algorithms was validated through 5-fold cross-validation.

4.1 Random Forest

4.1.1 Impurity Measures

Of the impurity measures available to determine which feature to split on, we studied three: Information Gain, Gini index, and misclassification error. Information gain is formulated as:

$$IG(T, a) = H(T) - H(T|a) \tag{1}$$

where a is the feature that is split, T is the rest of the tree, and $H(x)$ is the entropy of x.

The Gini index and misclassification rate are relatively simple, represented as $p(1-p)$ and p where p is the probablity of a misclassified example. Decision trees, when in the process of learning, will use these impurity measures and output different trees in some situations, varying results.

Dataset	Impurity Measure	Train Accuracy (%)	Test Accuracy (%)	Time (s)
Blobs	Misclassification	95.33	94.17	2.79
Blobs	Gini	95.42	93.83	2.81
Blobs	Entropy	95.46	94.17	2.94
Digits	Misclassification	100.00	61.20	30.18
Digits	Gini	100.00	70.80	33.59
Digits	Entropy	100.00	73.20	34.92

Table 2: Effects of impurity measure on classification accuracy and time elapsed.

4.1.2 Hyperparameters

The prominent numeric hyperparameter was the number of trees created to form the model. Each dataset was evaluated with decision forests of size 50, 100, and 150. This ended up, for some datasets, varying accuracy a fair amount, as seen in figure 2. Additionally, various stopping criteria hyperparameters had to tuned like a maximum tree depth or minimum examples for splitting on another feature.

4.1.3 Optimized Forests

Using the data collected from tuning the hyperparameters on a few of the datsets, we ran a single run on each dataset with the empirically optimal settings. Table 3 and Table 4 demonstrate those results.

4.2 SMO

We will explore the SMO learner in the coming weeks.

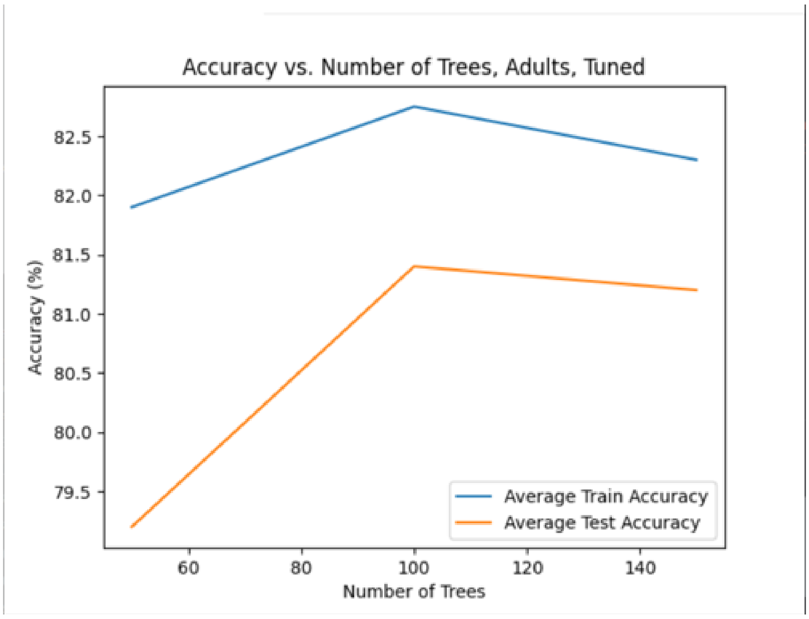


Figure 2: The effect of the number of trees on accuracy.

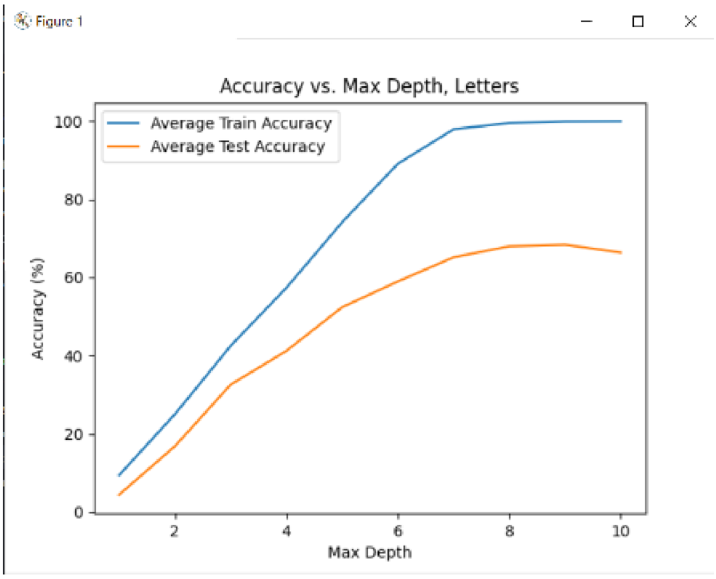
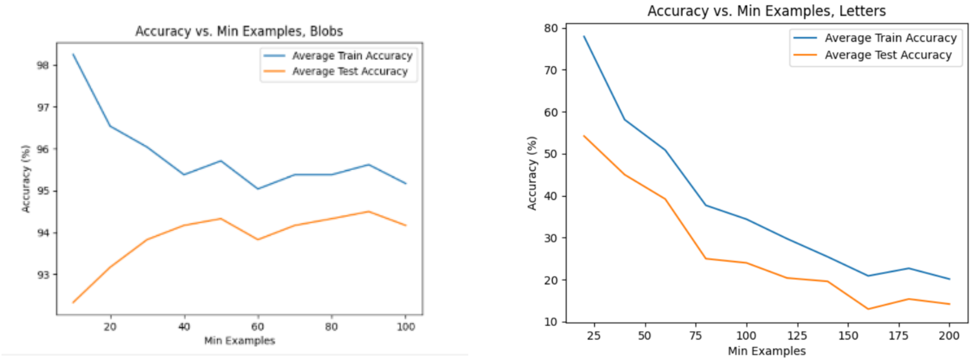


Figure 3: The effect of maximum Tree depth on accuracy.

5 Discussion

The three impurity measures performed rather predictably, as shown in Table 2. At worst, all three resulted in nearly identical accuracy splits. At best,



(a) Blobs

(b) Letters

Figure 4: The effects of minimum leaf examples on accuracy.

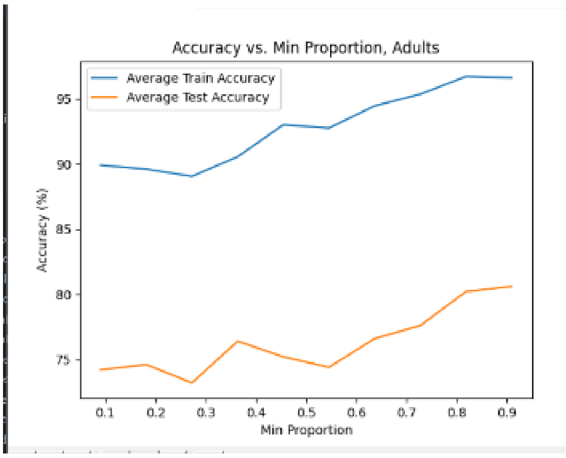


Figure 5: The effect of minimum subset proportion on accuracy.

the distinctions were clear and consistent: misclassification was the weakest measure and entropy was the strongest. The evidence might also suggest that the better measures come with higher time costs.

The hyperparameters need to be tuned for each and every model, as there does not seem to be a perfect rule which is applicable to each domain. For example, Table 3 and 4 the Random Forest performed best on the Blobs data when all three parameters were tuned to a specific value, whereas the best performance on the Spirals data occurred without setting any of them.

In general, test accuracy plateaus once a large enough maximum depth

Dataset	Train Accuracy	Test Accuracy	Time
	(%)	(%)	(s)
Adults	83.51	81.40	100.68
Blobs	95.88	94.00	3.61
Digits	100.00	65.60	35.63
Letters	99.90	67.80	47.29
Spirals	100.00	96.70	46.39
Zoo	100.00	97.00	0.18

Table 3: Results of single optimized run on each dataset.

Dataset	Examples	Trees	Max Depth	Min Examples	Min Prop
Adults	2500	100	3	0	1
Blobs	600	100	4	80	0.75
Digits	250	150	∞	0	1
Letters	500	100	9	0	1
Spirals	1000	100	∞	0	1
Zoo	101	100	∞	0	1

Table 4: Parameters used to optimize results. All used entropy as impurity measure.

has been reached, as shown in Figure 3. For some domains, it can be as shallow as 3 or 4 levels, and for others, as high as 9 or more. Figure 4 shows that the datasets were split in their reaction to increasing minimum leaf examples. The Blobs test accuracy peaked at around 80 minimum examples while the Letters accuracies only decayed exponentially. Most accuracies increased as the minimum subset proportion increased toward one, as in Figure 5.

Table 3 and Table 4 demonstrate that the Random Forests with no stopping criteria specified were able to learn the training data perfectly. In some cases, like the Spirals and Zoo datasets, this did not lead to a large drop off in testing set accuracy. In others, the testing accuracy was significantly lower.

6 Related Work

7 Conclusion

References

[1] Leo Breiman. *Machine Learning*, 45(1):5–32, 2001.