Random Forest and SMO Learners

Ethan Robards, Joe Shymanski

February 28, 2022

Abstract

This paper discusses the implementation of the Random Forest (RF) and Sequential Minimal Optimization (SMO) algorithms to learn an ensemble of Decision Trees (DTs) and an optimized Support Vector Machine (SVM), 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.

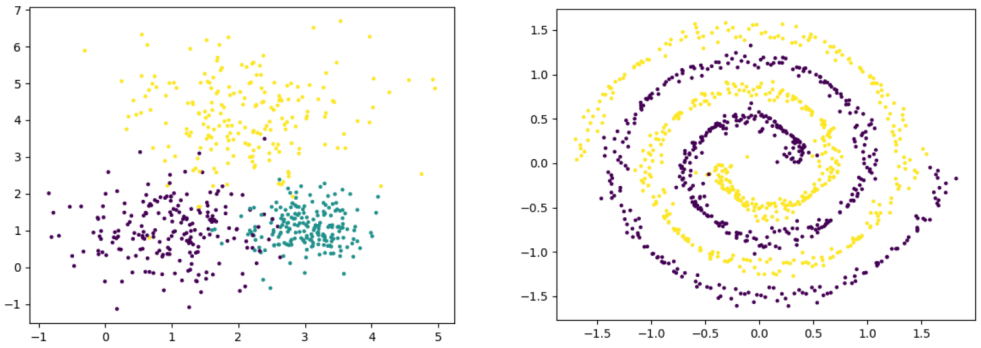
1 Introduction

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.

SMO is an algorithm which produces a single SVM which creates an optimal decision bound for binary classification. Since most real-world problems are not linearly separable due to noise or other various reasons, SMO employs soft margins to its boundary. Since SMO deals with binary classification and numeric features, the two artificial datasets will be used to guage its performace.

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 [5]. The Zoo data have a more physical feature set for classifying different species of animals [3]. 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 dataset and the Letters dataset both seek to classify handwritten and typeset images of alphanumerics, respectively [2, 7]. 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.



(a) Blobs

(b) Spirals

Figure 1: Plots for both auto-generated 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

Both models were created in Python 3 [8].

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

The SMO learner attempts to create an optimized SVM through iterative coordinate ascent, choosing a pair of coordinates at a time. This is to ensure an optimal model with soft margins, since most if not all of the problems are not linearly separable. The model uses three different kernel choices: simple inner product, the Gaussian kernel, and a custom or "polynomial" kernel. The Gaussian kernel is given by the equation:

$$K(x, z) = exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right) \tag{1}$$

where σ is a user-chosen parameter for standard deviation. As σ increases, the decision boundary becomes less strict. The polynomial kernel is as follows:

$$K(x, z) = \left(x^T z + c\right)^2 \tag{2}$$

where c is yet another parameter chosen by the user.

The learner returns the α_i's and the b parameters used for classification. These will be plugged into the predictor equation:

$$h(x) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \left(x_i \cdot x\right) + b\right) \tag{3}$$

where x is the input vector being predicted and the x_i's are the vectors from the training data.

4 Experimental Results

The accuracy for both algorithms was evaluated via 5-fold cross-validation.

4.1 Random Forest

The RF data was collected using a laptop with an Intel® Core™ i7-7700HQ CPU @ 2.80GHz.

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:

$$IG(T, a) = H(T) - H(T|a) \tag{4}$$

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. That variation is captured in Table 2.

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 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 Recision 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 be tuned like maximum Tree depth or minimum examples for splitting on another feature. These trends can be seen in Figure 3, Figure 4, and Figure 5.

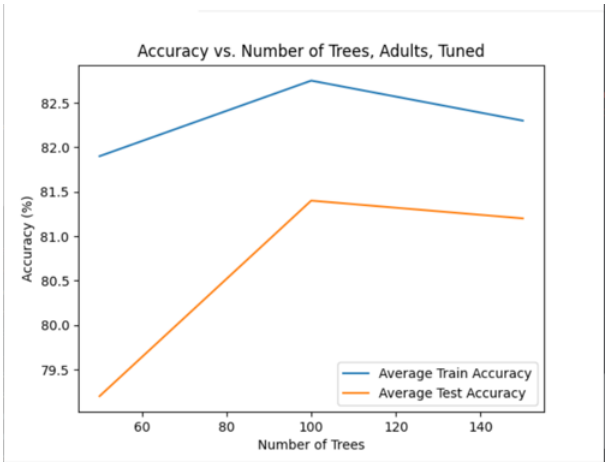


Figure 2: The effect of the number of Trees on accuracy.

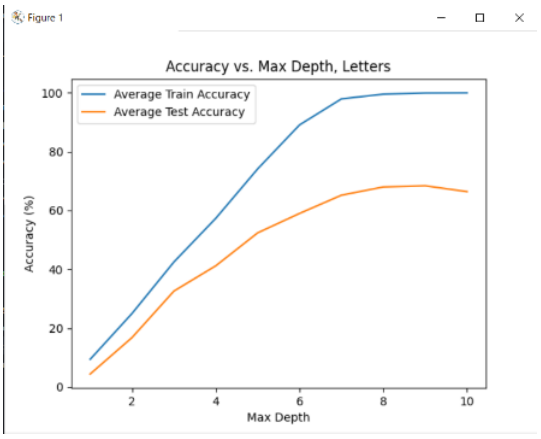
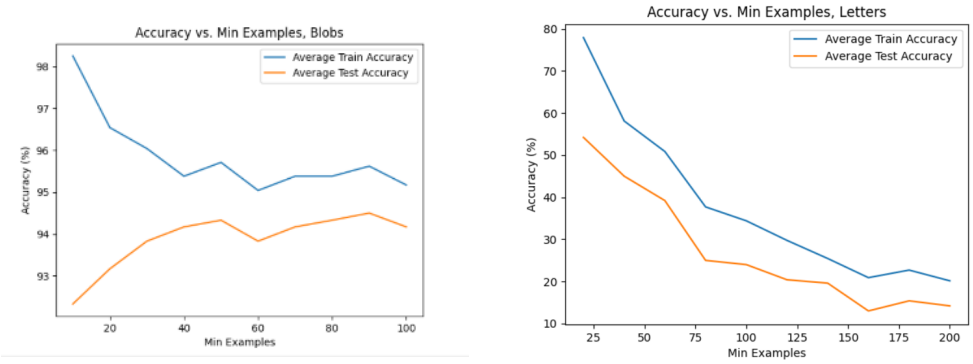


Figure 3: The effect of maximum Tree depth on accuracy.



(a) Blobs

(b) Letters

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

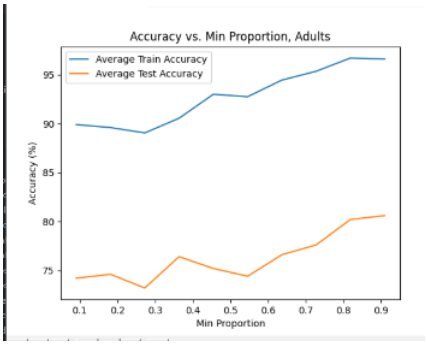


Figure 5: The effect of minimum subset proportion on accuracy.

4.1.3 Optimized Forests

Using the data collected from tuning the hyperparameters on a few of the datsets, we executed a single run of 5-fold cross-validation on each dataset with the empirically optimal settings. Table 3 and Table 4 demonstrate those results.

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 RF 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: RF parameters used to optimize results. All used entropy as impurity measure.

4.2 SMO

SVMs are used primarily for binary classification on datasets with numeric features. Since they create a single decision boundary, only two classes can be predicted at a time. The SVMs also assume feature values to be ordered, so many categorical attributes cannot be properly interpreted. Thus, the only datasets which will be experimented with are a new Blobs dataset and the original Spirals dataset.

The SMO algorithm calls for a check to ensure that the value of the second Lagrange multiplier (α_j) chosen to optimize is actually changed. To do so, it takes the difference of the new and old α_j's and determines if this value is close to zero. The threshold we used for this check was 10^{-5} for all experiments.

4.2.1 Kernels

Three different kernels were tested, two of which are given by Equation 1 and Equation 2. The performances of these kernels as C is varied are displayed in Figure 6. For the Gaussian and polynomial kernels, σ and c were both set to one.

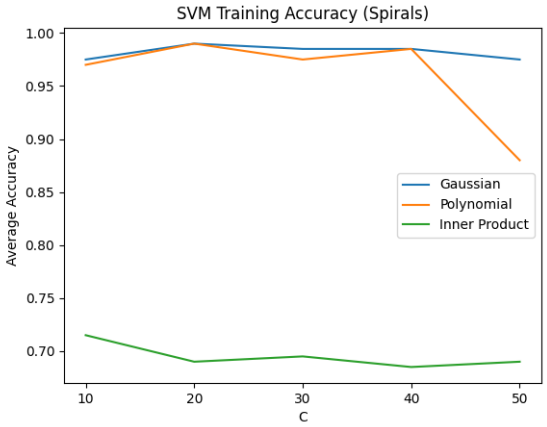


Figure 6: The effect of C and kernel on training accuracy.

4.2.2 Choice Heuristic

The process of choosing α_j has been completely random up to this point. However, there are heuristics available for choosing a better α_j, and the choice heuristic we decided to investigate is given by John Platt [6]. However, we ran into significant difficulties in getting any meaningful data from our implementation of Platt's pseudocode. The error cache would produce values that would overflow before the algorithm could complete. Thus, no data could be obtained to demonstrate its difference in performance.

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, 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 has been reached, as shown in Figure 3. For some domains, it can be as shallow as three or four levels, and for others, as high as nine 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.

The kernels chosen for the SMO algorithm made a huge difference in model accuracy, as displayed in Figure 6. The default inner product did not perform particuluarly well for any value of C. The Gaussian and polynomial kernels performed almost perfectly. In fact, even as the polynomial kernel decreased a bit as C approached 50, the Gaussian kernel stayed strong no matter its value.

Random Forest and SMO did not have many overlapping datatsets to compare their performances. However, they did share Spirals, and the SMO using a Gaussian kernel with $\sigma = 1$ outperformed the Random Forest with default parameters and no stopping criteria. This is not so surprising, since SVMs are hand-crafted to find decision boundaries in binary classification problems, and the kernel maps the featureset into different dimensions to solve non-linear problems. DTs, on the other hand, work well with discrete-

valued attributes and can be extended to handle the numerical features that are so natural to SVMs. DTs also do not individually combine features to form new ones like a kernel, though the hope is that a Random Forest may be able to mimic this.

6 Related Work

As mentioned before, Platt has provided a nice heuristic for choosing α_j in the SMO learner [6]. In the same paper, he provides other intelligent methods for speeding up SMO. One such suggestion was the use of an error cache, increasing simple lookups and reducing redundant error calculations. Another was to speed up the dot product calculation of two sparse vector by scanning them first. We attempted to implement his pseudocode as our extension of SMO, but it did not seem to perform very well.

Keerthi, Shevade, Bhattacharyya, and Murthy supplemented Platt's improvements with even more suggestions [4]. They provided two different modifications to Platt's SMO and reported that both were almost always more efficient than the original algorithm. Keerthi et al. used the same datasets as Platt, some of which are the same or very similar datasets used in this paper, such as Adults and Spirals.

7 Conclusion

In the end, RF seems to work best with the entropy measure of node impurity, though it comes with a slight increase in training time. Accuracy tends to increase as the number of DTs increases and plateaus around 100 Trees. The effects of the parameters can vary depending on the problem and need to be tuned on a case-by-case basis. It does seem that maximum depth could be a preferred stopping criterion to minimum examples.

The SMO algorithm worked best with the Gaussian and custom kernels, though they should be chosen and tuned according to the data. The simple SMO works well enough, albeit inefficiently. The alterations suggested by Platt and Keerthi et al. should provide significant speedup if implemented intelligently. The SVMs outperformed the DTs on the same problem, but not by a large margin.

References

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

[2] Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, November 2012.

[3] Richard Forsyth. UCI machine learning repository, 1990.

[4] S. Sathiya Keerthi, Shirish Krishnaj Shevade, Chiranjib Bhattacharyya, and Karuturi Radha Krishna Murthy. Improvements to platt's smo algorithm for svm classifier design. *Neural computation*, 13(3):637–649, 2001.

[5] Ronny Khavi and Barry Becker. UCI machine learning repository, 1996.

[6] John Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. 1998.

[7] David J. Slate. UCI machine learning repository, 1991.

[8] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.