Introduction to Intelligent Systems
Project2: Recycling used parts for fun and profit

Joseph Fuchs, Damien Crémilleux

November 21, 2014

1 Classifier Design

To solve this classification problem, we use two different techniques:

- Multi-layer perceptron
- Decision tree ensemble

The multi-layer perceptron has two input nodes, five hidden nodes, and four output nodes (for four classes), plus bias nodes (a 2-5-4 architecture), as suggested in the wording. This multi-layer perceptron is illustrated in Figure 1 page 2. The bias node are in yellow.

The multi-layer perceptron has two input nodes, five hidden nodes, and four output nodes (for four classes), plus bias nodes (a 2-5-4 architecture), as suggested in the wording. This multi-layer perceptron is illustrated in Figure 1 page 2. The bias node are in yellow.

The decision tree ensemble (aka. random forest) is a collection of decision trees, where the output of the forest is equal to the statistical mode of all the trees' outputs. Each tree is a sequence of paths which relate an input vector of continuous real-valued numbers to an output classification: Each tree will partition the set of real numbers for each input feature (element of the vector) one feature at a time until it associates the input with a classification guess. This type of random forest is illustrated in Figure 3 page 3, and the structure of the trees in this forest are illustrated in Figure 2 page 3. Note that the partition numbers on each internal edge of each tree, the ones used to choose which node to use next, are optimized for whichever training set was used to train that individual tree.

Decision tree ensemble and neural network are two methods of supervised learning. This means that each samples of the training data is a pair of an input object (in our case the value for the six-fold rotational symmetry and the eccentricity) and a desired output value (in our case, the class, bolt, nut, ring, scrap). However neural networks are not easily understandable. Neural networks are like a black box, we can not easily explain how decisions are made.

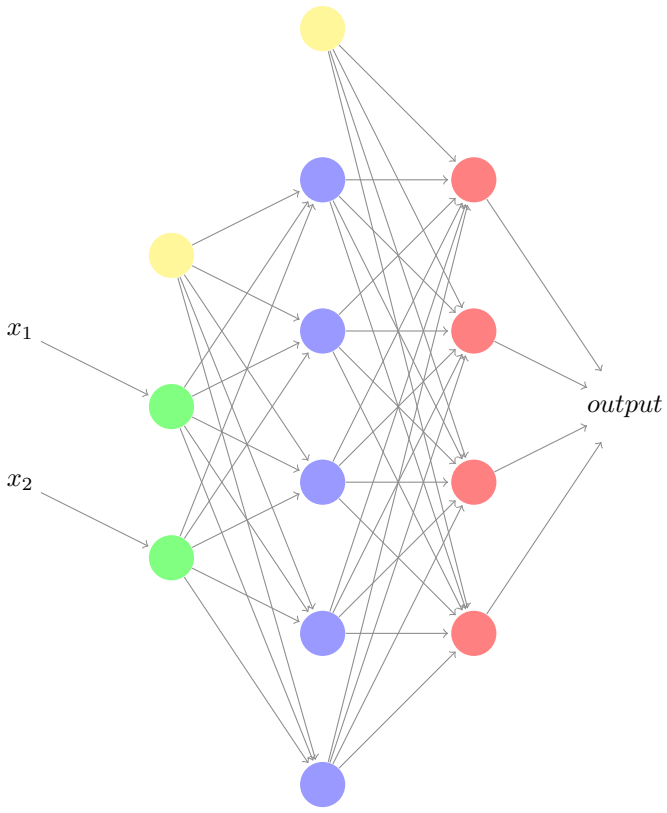


Figure 1: The architecture of the neural network used to solve this problem

Figure 2: Architecture of a Decision Tree

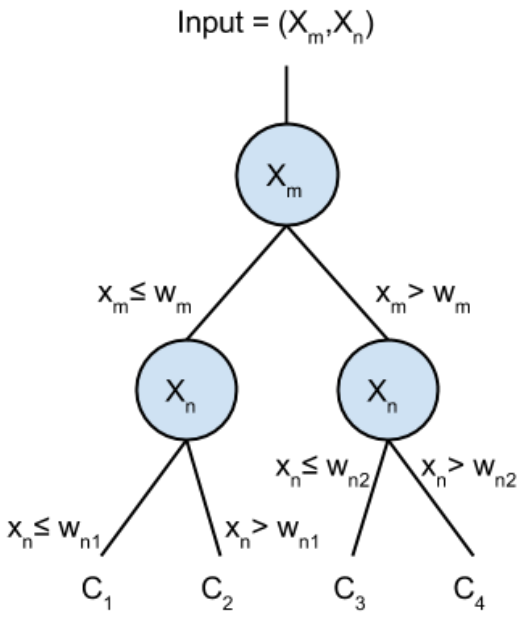
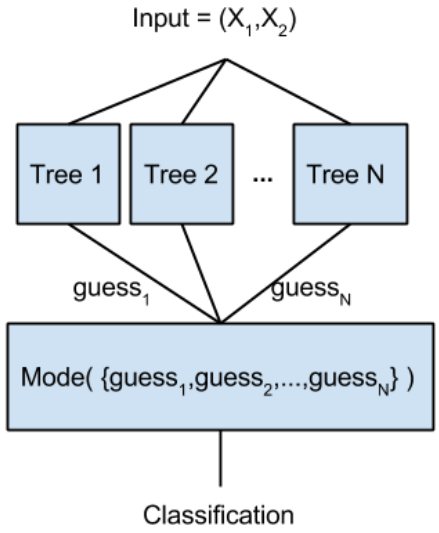


Figure 3: Architecture of a Random Forest of N Trees



On the contrary, it is easier to see how decisions are made in a decision tree ensemble because we can look at the trees and see how variables divide the data.

We expect the multi-layer perceptron to perform better. Several scientific articles[1] show that neural networks have better results than decision tree ensemble. However, the result heavily depends of the input data, this is not always the case. The main reason for this is because decision trees have a more difficult time sorting out data where the regions of one class overlap with another. Our decision trees will only partition the input space into rectangular regions, and even if we gather many trees, the precision required to sort out overlapping regions will be outvoted by the majority. Multi-layer perceptrons, on the other hand, can divide regions on the input space more precisely.

2 Data sets

The training data set is shown in Figure 4 and the test data set is shown in Figure 5.

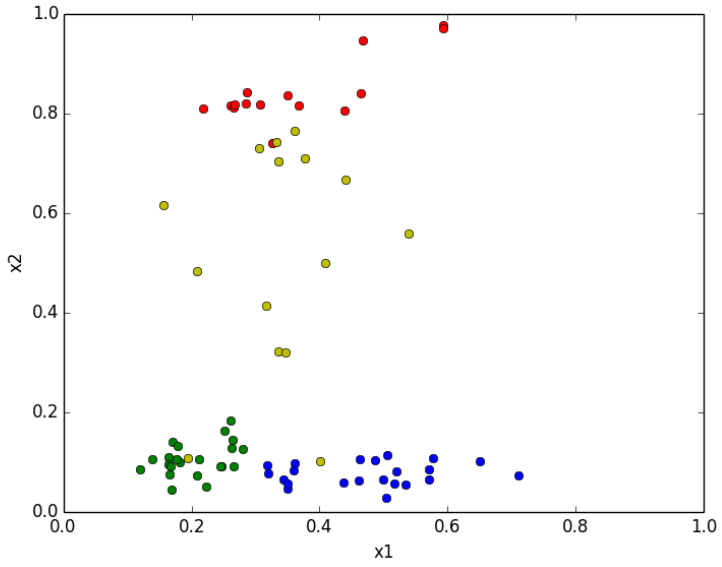


Figure 4: Training data set

The red dots are bolts, the blue dots are nuts, the green dots are rings and the yellow ones are scraps. We can see that on the test set, the dots are not mixed. This is not the case for the training data set. A bolt is mixed with scraps, and scraps are mixed with nuts and rings. This will lead to some errors

[1] An example in bio-informatic: `http://www.csee.usf.edu/~rbanfiel/papers/NNvsDT.pdf`

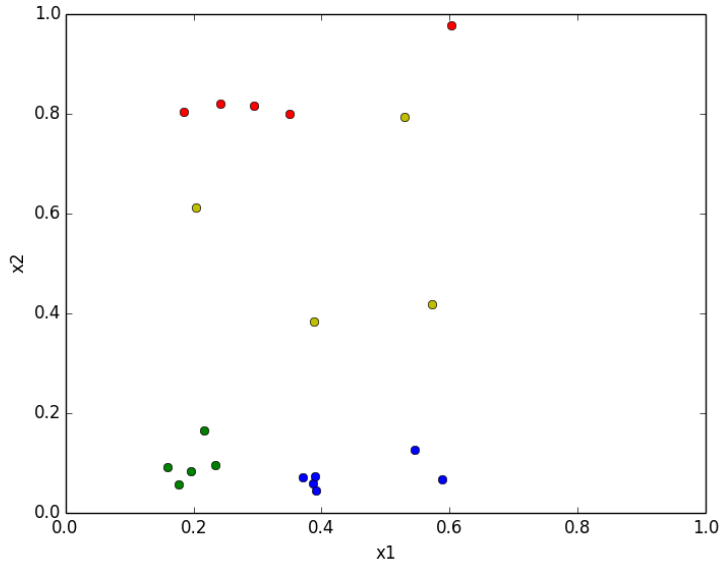


Figure 5: Test data set

of classification for the training data set, but because the different classes are not mixed in the test data set, we can hope that the final classifier will be efficient.

3 Results

3.1 Multi-layer perceptron

Epoch	0	10	100	1000	10000
Recognition rate	0.2	0.25	0.65	1	1
Profit	-0.6	-0.8	1.03	2.03	2.03
Confusion matrix	[0, 0, 0, 0] [0, 0, 0, 0] [0, 0, 0, 0] [5, 6, 5, 4]	[0, 0, 0, 0] [0, 0, 0, 0] [5, 6, 5, 4] [0, 0, 0, 0]	[5, 0, 0, 2] [0, 2, 0, 1] [0, 4, 5, 0] [0, 0, 0, 1]	[5, 0, 0, 0] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 4]	[5, 0, 0, 0] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 4]

3.2 Decision tree ensemble

Due to the randomness in the construction of trees in the random forest, to achieve a more accurate set of data, we calculated the average (mean) of five distinct constructed random forests.

Trees	1	10	100	200
Recognition rate (mean)	0.82	0.83	0.85	0.87
Profit (mean)	1.43	1.78	1.87	1.87
Confusion matrix Forest 1	[3, 0, 0, 1] [0, 6, 0, 0] [0, 0, 5, 0] [2, 0, 0, 3]	[5, 0, 0, 4] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 0]	[5, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 1]	[5, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 1]
Confusion matrix Forest 2	[3, 0, 0, 1] [0, 6, 0, 0] [0, 0, 5, 0] [2, 0, 0, 3]	[5, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 2]	[5, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 2]	[5, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 2]
Confusion matrix Forest 3	[3, 0, 0, 1] [0, 6, 0, 0] [0, 0, 5, 0] [2, 0, 0, 3]	[5, 0, 0, 4] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 0]	[5, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 1]	[5, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 1]
Confusion matrix Forest 4	[1, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [4, 0, 0, 2]	[3, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [2, 0, 0, 2]	[4, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [1, 0, 0, 1]	[4, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [1, 0, 0, 2]
Confusion matrix Forest 5	[3, 0, 0, 1] [0, 6, 0, 0] [0, 0, 5, 0] [2, 0, 0, 3]	[4, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [1, 0, 0, 2]	[5, 0, 0, 3] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 1]	[5, 0, 0, 2] [0, 6, 0, 0] [0, 0, 5, 0] [0, 0, 0, 2]

3.3 Classification regions

The evolution of the classification regions produced by the multi-layer perceptron is shown from Figure 6 to Figure 10.

The evolution of the classification regions produced by the random forest is shown from Figure 11 to Figure 14.

3.4 Learning curve

The learning curve image for the neural network is shown in Figure 15.

And the learning curve image for the decision tree ensemble is shown in Figure 16. We defined the error or a forest to be the fraction of total tree votes that were not for the correct class.

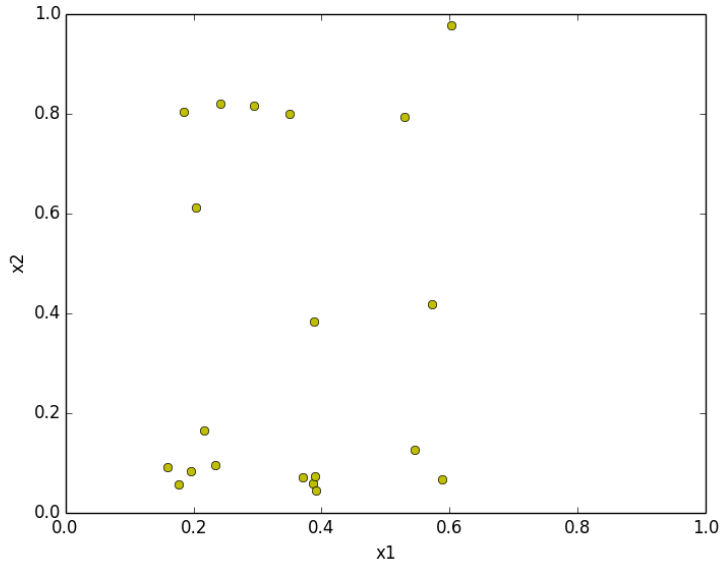


Figure 6: Classification produced by the multi-layer perceptron after 0 epoch

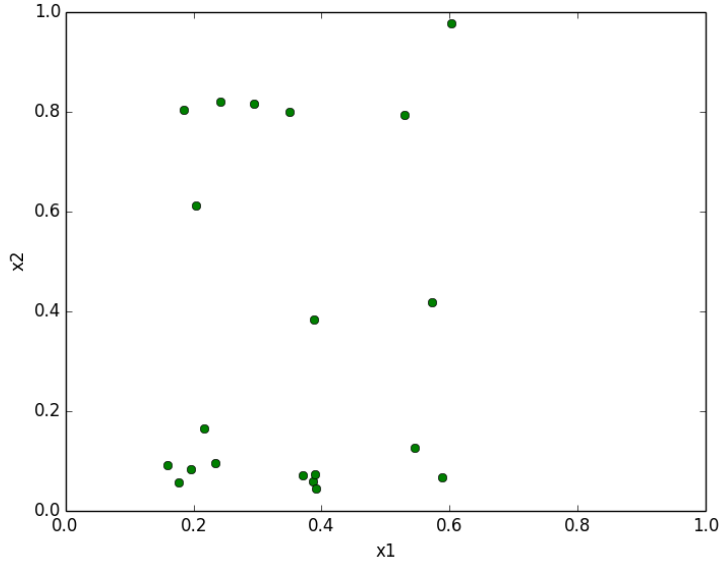


Figure 7: Classification produced by the multi-layer perceptron after 10 epochs

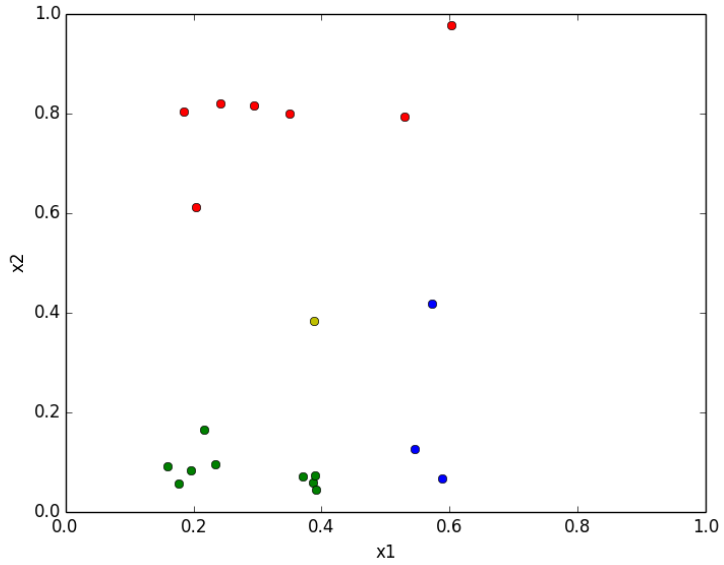


Figure 8: Classification produced by the multi-layer perceptron after 100 epochs

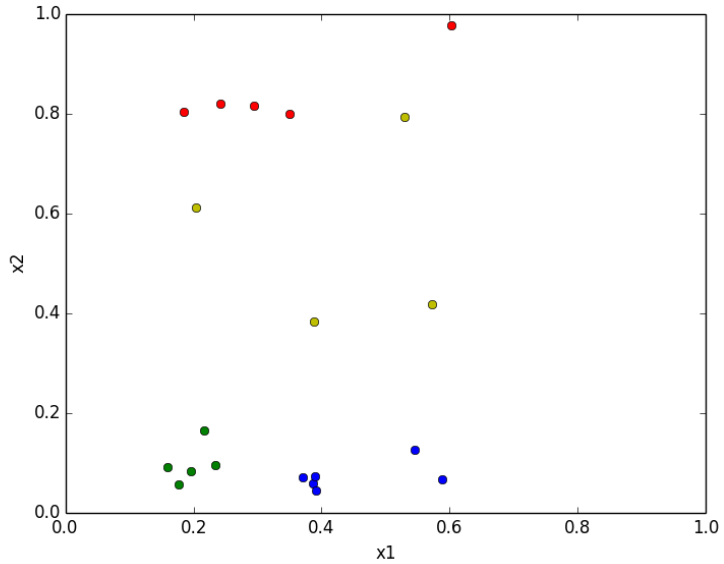


Figure 9: Classification produced by the multi-layer perceptron after 1000 epochs

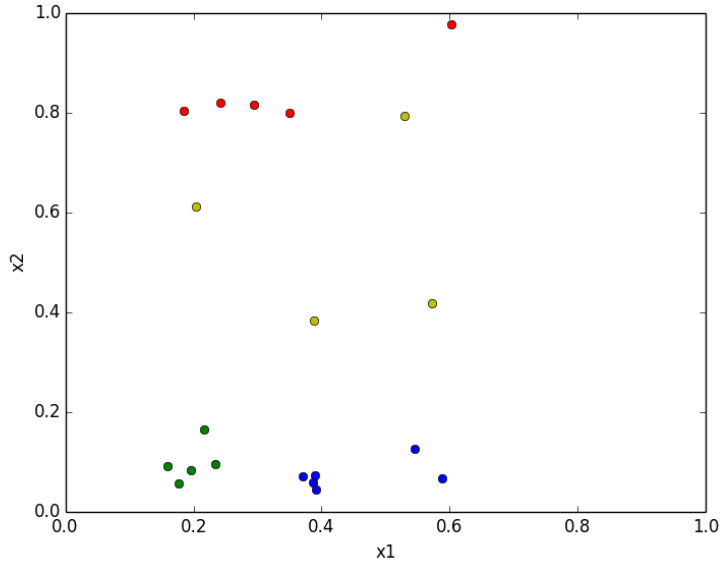


Figure 10: Classification produced by the multi-layer perceptron after 10000 epochs

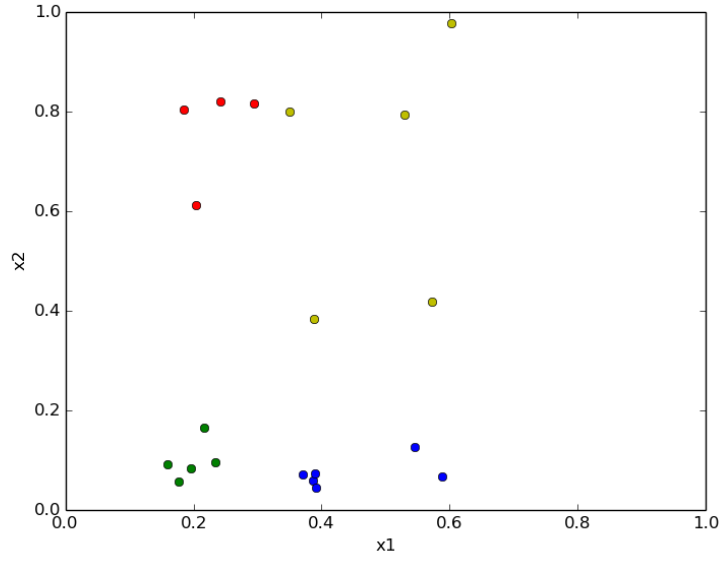


Figure 11: Classification produced by the random forest after 1 tree has been added

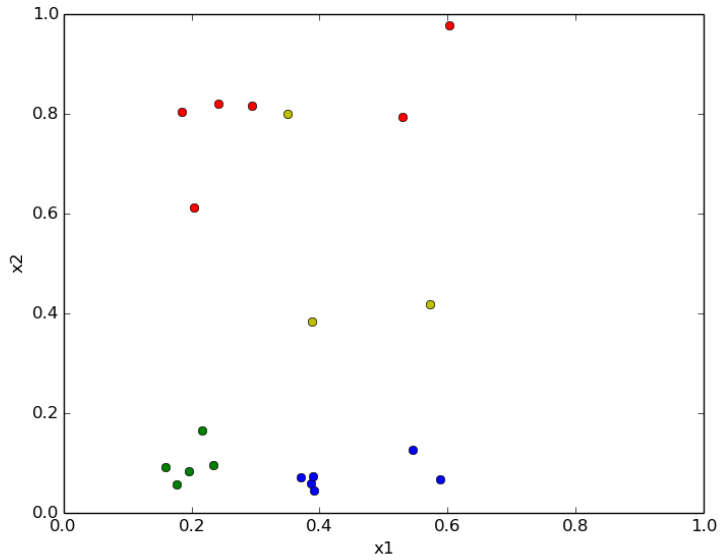


Figure 12: Classification produced by the random forest after 10 trees has been added

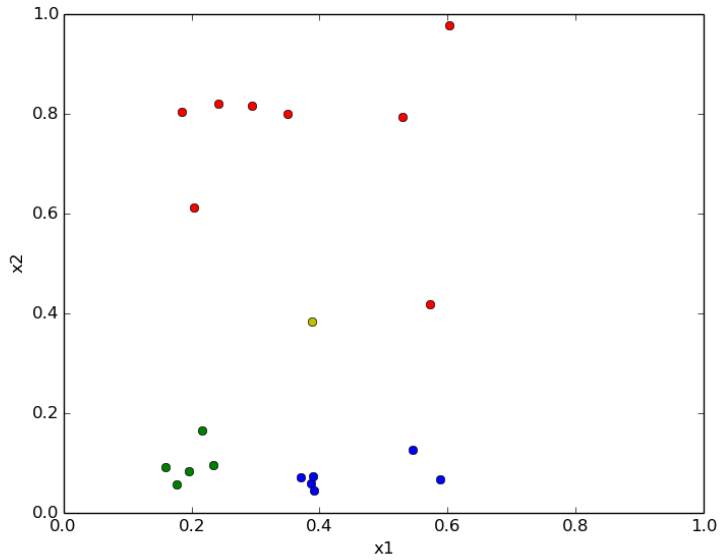


Figure 13: Classification produced by the random forest after 100 trees has been added

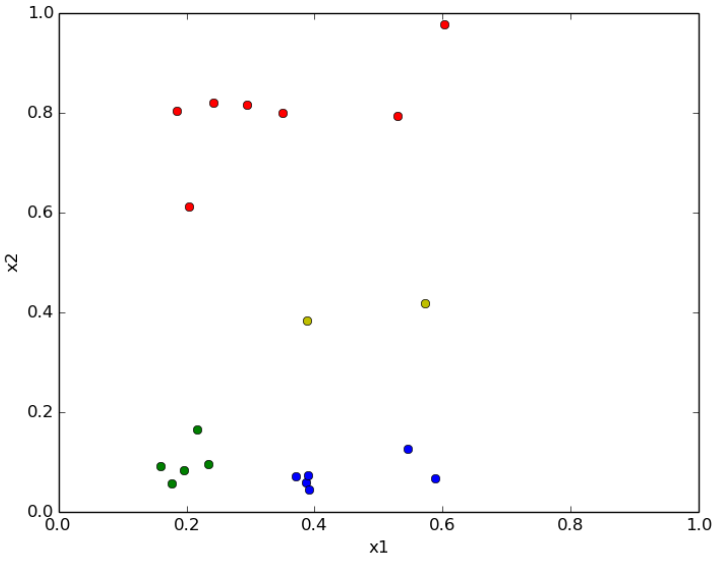


Figure 14: Classification produced by the random forest after 200 trees has been added

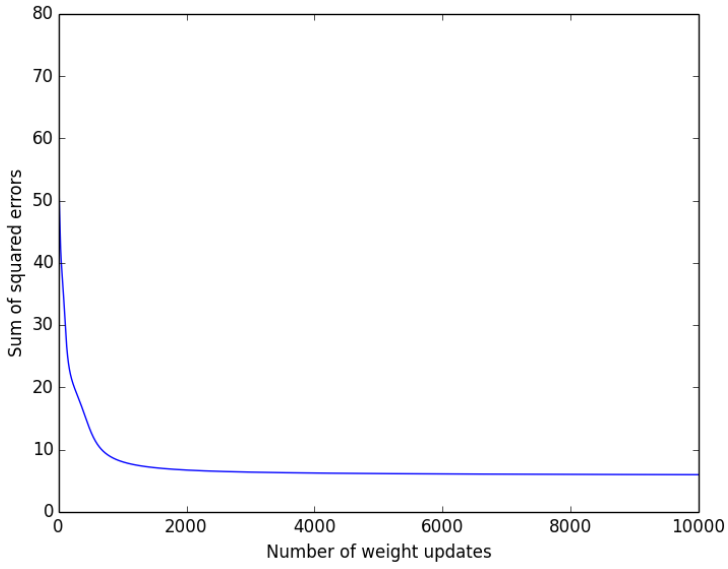


Figure 15: Learning curve for the multi-layer perceptron

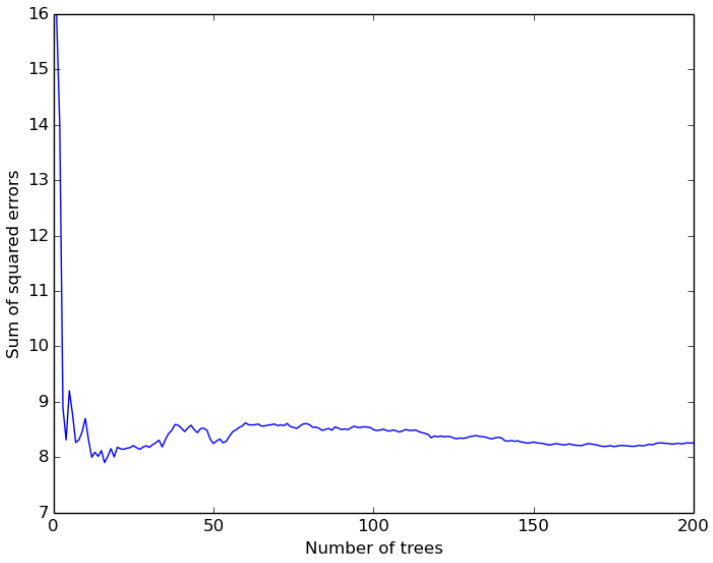


Figure 16: Learning curve for the decision tree ensemble

4 Discussion

With the data from the Section 3, we can see that the multi-layer perceptron correctly classified all the samples from the test data after 1000 epochs. The learning curve shows us that the multi-layer perceptron learns very quickly during the early epochs, and then reaches a plateau after 2000 epochs. We can see that first the multi-layer perceptron classifies all the samples as scraps. Then, after 10 epochs, all the samples are classified as rings. After 100 epochs, the majority of samples are correctly classified, however the class boundaries are still imperfect. This is why the recognition rate is only 0.65 at this point. Finally, after 1000 epochs, the class boundaries are correct.

Secondly, for the random forest, we can see that it classified most but not all samples from the test data after 200 trees. The learning curve shows us that the random forest also learns quickly and reaches a plateau after the amount of trees in the forest grows.

Even though the recognition rate of the forest is high and it converges quickly as trees are added, the recognition rate does not converge to 100% like it did in the multi-layer perceptron. This follows our hypothesis. In order to achieve more accurate trees, we would need to add more branches to each tree node so the partitioning could become more than 2 for each sample feature. This would likely sacrifice a large amount of computation time for optimizing the branch numbers from each tree node. We also see that the forest we used learns faster, but not perfectly, whereas the MLP learns more slowly, but can become more accurate in the long-run.