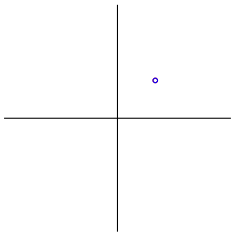
HOMEWORK 2

Daniel Szabo
9074769625

1 Decision Tree Questions

1. The information gain of any split for a node that has every label the same would be zero. This matches one of the stopping criteria, and therefore the node should be a leaf. In particular the algorithm we saw in class would not even find any candidate splits.

2. If the same features have different label values, the algorithm would simply make a leaf and stop. If we were to force a split, the different labels would still be in the same subset of the training set, and therefore the algorithm would continue indefinitely. This experiment could be done with any number (> 1) of training points, in particular 2:



   Here the single point in the dataset appears twice, once with label '1' and once with the '0' label, although this is hard to plot.

3. The candidate splits for feature 0 and their corresponding information gain ratios:
   Cut 0.0 has gain ratio 0.10051807676021828
   Cut 0.05 has gain ratio 0.10051807676021828

   Note that a cut at 0.0 would be the same as $0 + \epsilon$ for $\epsilon > 0$ small, as the algorithm breaks ties downwards. This is why the above gain ratios are the same.
   The splits and ratios for feature 1 are:
   Cut -1.5 has gain ratio 0.10051807676021828
   Cut -0.5 has gain ratio 0.055953759631263526
   Cut 5.5 has gain ratio 0.23609960614360798
   Cut 6.5 has gain ratio 0.055953759631263526
   Cut 7.5 has gain ratio 0.4301569161309808

4. The (simplified) boolean function this tree represents is $f(x_1, x_2) = (x_2 > 2.0 \vee x_1 > 1.0)$, where a label 1 corresponds to true. The tree is
   1. if $x_1 \leq 1.0$:
   2. if $x_2 \leq 2.0$:
   3. $y = 0$
   4. else:
   5. $y = 1$
   6. else:
   7. $y = 1$

5. (a) The tree is very simple:
   1. if $x_2 \leq 0.200777$:
   2. $y = 0$
   3. else:
   4. $y = 1$

(b) There is just a horizontal line at $x_2 = 0.200777$; if the point is above this line, it is classified as 1, if it's below, the label is 0. The space is split perfectly in two by a single separating hyperplane.

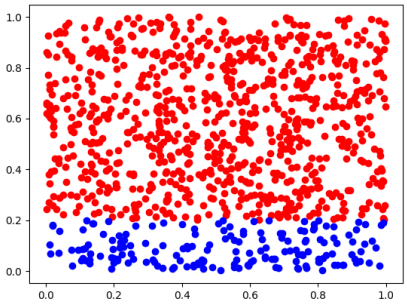
(c) The decision tree is very large and can't really be viewed here... nonetheless here are the first few lines:

1. If $x_2 \leq 0.9182275$:
2. If $x_1 \leq 0.53287$:
3. If $x_2 \leq 0.6911515$:
4. If $x_2 \leq 0.5338315$:
5. y=0.0
6. Else:
7. If $x_2 \leq 0.5362020000000001$:
8. y=1.0
9. Else:
10. If $x_2 \leq 0.5533045000000001$:
11. y=0.0
12. Else:
13. If $x_2 \leq 0.5550665$:
14. y=1.0

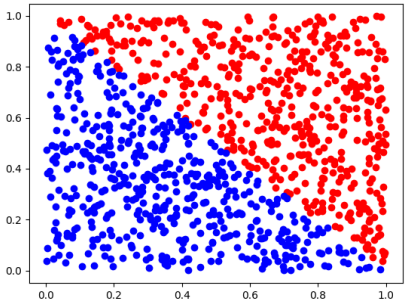
⋮

(d) It seems the points do not match anything our decision tree can decide. They may be scattered completely randomly, which is why the tree is so large. We cannot be certain without visualization.

6. The plot for `D1.txt`:

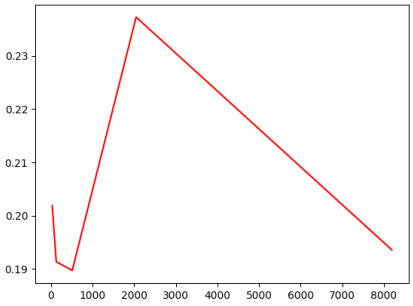


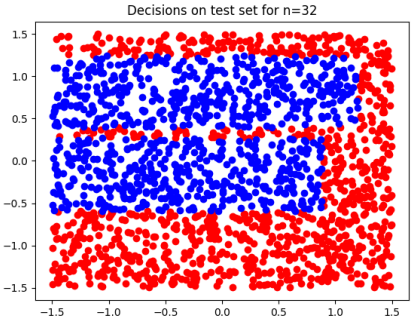
And for `D2.txt`:



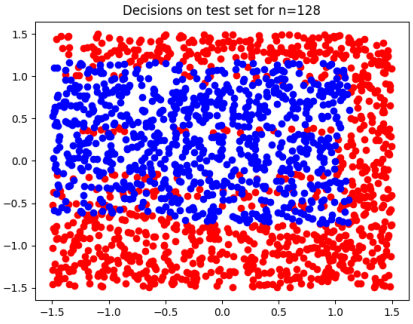
This explains why D2 was so incredibly large, as the decision tree can only create horizontal and vertical boundaries, which cannot match the true decision boundary. We have a hypothesis space bias towards these specific vertical or horizontal decision boundaries, rather than arbitrary linear separators.

7. The learning curve is:





$n = 32$, number of nodes=7, $err_n = 0.20188053097345138$



$n = 128$, number of nodes=29, $err_n = 0.1913716814159292$

2 sklearn

$n = 32$, number of nodes $= 5$, $err_n = 0.20630530973451322$
$n = 128$, number of nodes $= 10$, $err_n = 0.08683628318584069$
$n = 512$, number of nodes $= 36$, $err_n = 0.04646017699115046$
$n = 2048$, number of nodes $= 62$, $err_n = 0.02488938053097345$
$n = 8192$, number of nodes $= 125$, $err_n = 0.014380530973451378$
The (impressive) learning curve is in Figure 9:

Figure 6: $n = 512$, number of nodes=93, $err_n = 0.18971238938053092$

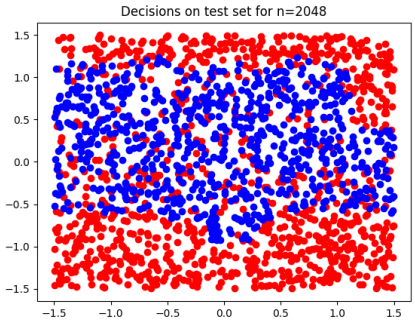


Figure 7: $n = 2048$, number of nodes=516, $err_n = 0.2372787610619469$

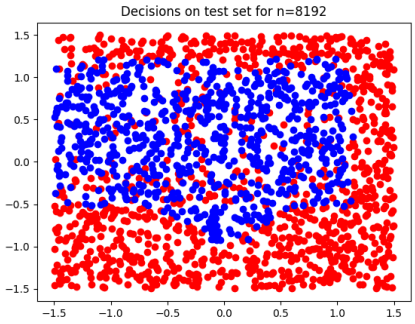


Figure 8: $n = 8192$, number of nodes=1507, $err_n = 0.19358407079646023$

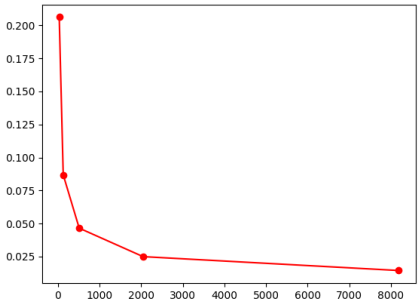


Figure 9: Learning curve for sklearn's decision tree algorithm

3 Lagrange Interpolation

I sampled 100 points uniformly at random from $[0, 20)$ and computed the training and testing MSEs of Lagrange interpolation to get
Train error (MSE): 0.0
Test error (MSE): 67646250994765.36
Clearly, Lagrange interpolation does not do well on test data, as it has very large coefficients and only performs well at precisely the training points.
Even when we add very small error terms, the error is incredibly huge:
Error for $\epsilon \sim \mathcal{N}(0, 0)^{100}$: 0.0
Error for $\epsilon \sim \mathcal{N}(0, 10^{-12})^{100}$: 735485451224.1586
Error for $\epsilon \sim \mathcal{N}(0, 10^{-8})^{100}$: 4540211115700.72
Error for $\epsilon \sim \mathcal{N}(0, 10^{-6})^{100}$: 61576526883137.39
Error for $\epsilon \sim \mathcal{N}(0, 0.01)^{100}$: $1.2972225357016978e + 18$
Fortunately we see that the error is growing at a polynomial rate, just that the polynomial is very large.