Christopher Hammack
CSCE970
Homework 2 Preliminary Results
March 25, 2003

1 Decision Tree Implementation

During this process of creating a decision tree implementation, three different learning tree algorithms were created. Each of the decision tree algorithms uses the concept of information gain (Equation 1 & 2) to choose the most appropriate choice for the next node in the decision tree. The first algorithm, referred to as the *naive* version, simply partitions each dimension in halves, and chooses the dimension with maximum information gain. The second algorithm *iterative* searches for the best point to draw the decision line using specific increments. For the purpose of this preliminary paper, to simplify the results, only one parameter is given, with 100 equal possible partitions comprising the search space. Finally, the algorithm *boundary* projects the points in a particular dimension on a line, and scans along the line looking for changes in classifications. Each change of classification is in the search space for the maximum information gain.

$$ent(S) = -\sum_i \frac{N_i}{N} \log_2 \frac{N_i}{N} \tag{1}$$

$$informationGain(S, a) = ent(S) - (\frac{S_a}{S} Ent(S_a) + \frac{S'_a}{S} Ent(S'_a)) \tag{2}$$

In general, the naive version performs surprisingly well. This is mostly just luck however that the datasets have classes which are mostly separated with a considerable distance between them, The iterative version generally performs extremely well, aside from dataset 2, where it classified everything as class 1. However, this version is generally very slow because of the amount of time it takes to check each of 100 possible decision line locations in n dimensions. The boundary version generally provides approximately the same performance as the iterative version, with a smaller run time in many cases. However, the boundary version could be quite slow as well when there were a lot of class changes in any particular dimension. This could be seen in dataset 6.

Figure 1: Dataset 1

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Total	Error Rate (95% Confidence)
Naive	0.74	0.9994	0.9827	0.9944	0.0066 ± 0.0021
Iterative	1.75	0.9989	0.9847	0.9946	0.0064 ± 0.0020
Boundary	1.23	0.9994	0.9707	0.9908	0.0092 ± 0.0026

Figure 2: Dataset 2

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Total	Error Rate (95% Confidence)
Naive	0.9	0.9830	0.6660	0.919	0.081 ± 0.0075
Iterative	4	1	0	0.8	0.2 ± 0.0111
Boundary	4.2	0.9328	0.6510	0.8764	0.1246 ± 0.0091

Figure 3: Dataset 3

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Total	Error Rate (95% Confidence)
Naive	0.8	0.9928	0.9872	0.99	0.01 ± 0.0028
Iterative	3.10	0.9936	0.9872	0.9904	0.0096 ± 0.0027
Boundary	1.4	0.9904	0.9852	0.9878	0.0122 ± 0.0030

Figure 4: Dataset 4

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Total	Error Rate (95% Confidence)
Naive	0.8	0.9993	1	0.9996	0.0004 ± 0.0006
Iterative	1.7	0.9993	0.999	0.999	0.001 ± 0.0009
Boundary	1.1	1	1	1	0.0 ± 0.0

Figure 5: Dataset 5

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Total	Error Rate (95% Confidence)
Naive	1	0.9976	1	0.9986	0.0014 ± 0.0010
Iterative	1.9	0.995	1	0.997	0.003 ± 0.0015
Boundary	1.7	0.9993	1	0.9996	0.0004 ± 0.0006

Figure 6: Dataset 6

Algorithm	Run time (sec)	Correct Rate Class 1	Class 2	Class 3	Class 4	Total	Error Rate (95% Confidence)
Naive	1	0.9157	0.9086	0.859	0.994	0.9268	0.0732 ± 0.0072
Iterative	5.3	0.9476	0.9743	0.9143	0.99	0.9552	0.0448 ± 0.0057
Boundary	8.2	0.94095	0.8428	0.88	0.99	0.936	0.064 ± 0.0068