Assignment 4 - Decision Trees

Nils Herde & Aleksander Skraastad

April 21, 2016

1 Part A - Results

Running with random importance function, the program yielded the following correctness score:

```
Random Score: 85.71
```

The following graph visualization of the tree has been generated from *PHPSyntaxTree* [1]

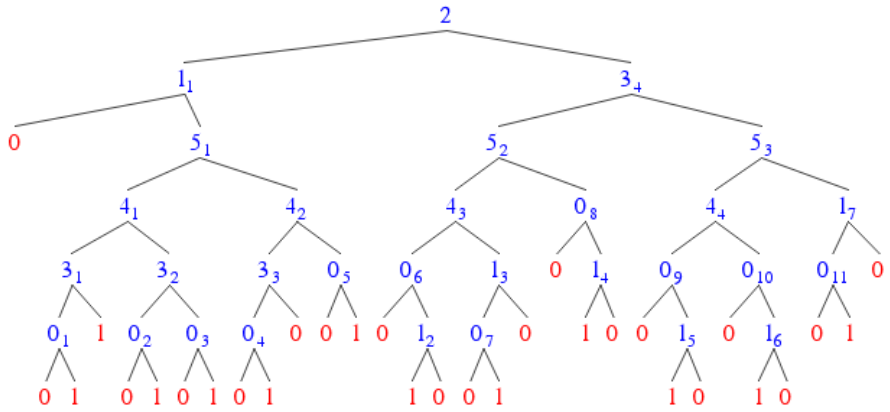

Figure 1: Random importance function

Running another run yields a test score of 92.86. For the most time, the random importance function will create a very large tree, with below 100.00 classification rate. Below is an exerpt of 5 consecutive runs with random:

```
Score: 92.86
Score: 85.71
Score: 85.71
Score: 100.00
Score: 85.71
```

Figure 2 shows the tree when using the information gain importance function. It immediately identifies an appropriate attribute to split the dataset on, and settles.

```
Information Gain Score: 100.00
```

Running the algorithm again produces the same tree, split on the same attribute.

[1] `http://ironcreek.net/phpsyntaxtree/`

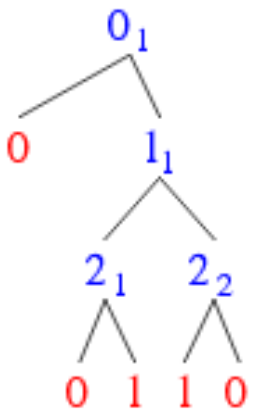

Figure 2: Information gain importance function

5 consequtive runs with the information gain function yields the following results:

```
Score: 100.00
Score: 100.00
Score: 100.00
Score: 100.00
Score: 100.00
```

2 Part B - Discussion

Based on these results, it is easy to conclude that the information gain function is superior. It produces the same tiny tree each time, compared to the very unstable random tree generation. When using the random function, there is not a guarantee of being able to correctly classify all objects either.

As stated previously, running the random over and over again produces greatly different results. The information gain learner uses entropy values to select which attribute to split on, and since the data set never changes, the entire operation is deterministic, and will produce the same results each time.