MDL Assignment-4: Decision Trees

Jaidev Shriram (2018101012)

April 2020

1 Introduction

This assignment requires the creation of a decision tree when given a table of examples which are defined by a set of examples. The attributes may take multiple values - and in my case, is a continuous distribution of real numbers and not a discrete distribution.

This task was solved using the greedy method of decision making. In this algorithm, we choose the local maxima to build a tree.

2 Data Set

The examples given are:

It is immediately clear that these examples consist purely of continuous values. Hence, when making a decision tree, it is not practical to partition the attributes based on value but rather ranges. For instance, we may choose to divide the examples based on attribute *speed* with the condition $value <= 50$.

Horizontal Angle(degree)	Distance(m)	Wind Speed(mph)	Kill
1.5	450	220	N
4.5	520	-120	N
3	490	120	Y
5.5	530	117	Y
3.2	470	-170	N
5.2	505	-90	Y
1.85	465	120	Y
4.8	517	147	Y
1.7	430	-100	Y

Table 1: Initial Data Set

3 Algorithm Used

The algorithm used is the greedy algorithm specified in the textbook *AI: A Modern Approach*.

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree ← a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    subtree ← DECISION-TREE-LEARNING(exs, $attributes - A$, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Figure 1: Decision Tree Algorithm

The subsequent section will explain how this algorithm was used.

4 Approach

Since all attributes were continuous, it wasn't practical to use trial and error for this task. This is because even when trying to make a choice about the attribute that must be used at a node, the importance of that attribute is limited by the choice of partition that we make for that attribute.

For instance, *speed* $<= 50$ may yield importance / gain of 0.5. Another attribute may have a higher gain and will be prioritized in the greedy algorithm. However, for the test *speed* $<= 100$, the gain may be 0.9 which is higher than our previous best attribute! Hence, it is critical that as part of our greedy choice, we brute force through all possible partitions of the example set and choose the best possible partition to represent the gain of that attribute.

Hence, I coded the algorithm and fed as input this data set. At every call to the *Decision Tree Algorithm*, it chooses the attribute with the best gain. The gain for any attribute is once again calculated by choosing the best gain from all the possible partitions of the examples based on that attribute.

```
def importance(attribute, examples, parent_examples):

    examples = sort(attribute, examples)
    max_i = -100
    max_gain = -100

    for i in range(1, len(examples)):

        positive = [0, 0]
        negative = [0, 0]
```

```
        for j in range(len(examples)):
            if float(examples[j][attribute]) < float(examples[i][attribute]):
                if examples[j]["kill"]:
                    positive[0] += 1
                else:
                    negative[0] += 1
            else:
                if examples[j]["kill"]:
                    positive[1] += 1
                else:
                    negative[1] += 1

        remainder = calc_remainder(positive, negative, parent_examples)
        gain = calc_parent_entropy(parent_examples) - remainder

        if gain > max_gain:
            max_gain = gain
            max_i = i

    return max_gain, max_i
```

Listing 1: Brute Force Importance Calculation

The above code is a simple illustration of how the algorithm works. It clearly iterates through all the examples, sets each one as the split for the data set and then computes the gain for the data set.

5 Trace

Note that entropy (B) has been shown for every level. This is calculated as:

$$B(q) = -q\log(q) - (1-q)\log(1-q) \tag{1}$$

Gain is calculated as:

$$Gain(Attribute) = B(ParentExamples) - Remainder(Attribute) \tag{2}$$

Remainder is calculated as:

$$Remainder(Attribute) = \sum \frac{(p_k + n_k)}{p+n} B(\frac{p_k}{p_k + n_k}) \tag{3}$$

Since *Gain* is a relatively simply equation, the calculation for gain has not been shown. The remainder values along with the entropy values should be enough to calculate the gain.

Note that *Set i* refers to the ith child after splitting the data set according to a split condition.

We can now attempt to determine which attribute has the most importance / maximum gain:

5.1 Level 1

The entropy of this set of examples is:

$$B(\frac{6}{9}) = \frac{6}{9}\log\frac{6}{9} + \frac{3}{9}\log\frac{3}{9} = 0.9182958340544896 \tag{4}$$

5.1.1 Angle

Best Gain is for $angle < 4.8$ and $angle >= 4.8$

Set 0: Kill (Yes): 3 Kill (No): 3
Set 1: Kill (Yes): 3 Kill (No): 0

Remainder: $\frac{6}{9} * B(\frac{3}{6}) + \frac{3}{9} * B(\frac{3}{3}) = 0.6666666666666666$

Remainder: 0.6666666666666666
Gain: 0.2516291673878229

The remaining split points and the corresponding gains are:

- $angle < 1.7$ and $angle >= 1.7$ **Remainder**: 0.7211361106303402 **Gain**: 0.19715972342414934
- $angle < 1.85$ and $angle >= 1.85$ **Remainder**: 0.8935382199962686 **Gain**: 0.024757614058220967
- $angle < 3.0$ and $angle >= 3.0$ **Remainder**: 0.9182958340544896 **Gain**: 0.0
- $angle < 3.2$ and $angle >= 3.2$ **Remainder**: 0.8999850522344305 **Gain**: 0.018310781820059074
- $angle < 4.5$ and $angle >= 4.5$ **Remainder**: 0.8999850522344305 **Gain**: 0.018310781820059074
- $angle < 4.8$ and $angle >= 4.8$ **Remainder**: 0.6666666666666666 **Gain**: 0.2516291673878229
- $angle < 5.2$ and $angle >= 5.2$ **Remainder**: 0.7662885502488623 **Gain**: 0.15200728380562722
- $angle < 5.5$ and $angle >= 5.5$ **Remainder**: 0.8483857803777466 **Gain**: 0.06991005367674297

5.1.2 Distance

Best Gain is for $dist < 465.0$ and $dist >= 465.0$
Set 0: Kill (Yes): 2 Kill (No): 2
Set 1: Kill (Yes): 4 Kill (No): 1

Remainder: $\frac{4}{9} * B(\frac{2}{4}) + \frac{5}{9} * B(\frac{4}{5}) = 0.8455156082707569$

Remainder: 0.8455156082707569
Gain: 0.07278022578373267

The remaining split points and the corresponding gains are:

- *dist* < 450.0 and *dist* $>= 450.0$ **Remainder**: 0.8483857803777466 **Gain**: 0.06991005367674297
- *dist* < 465.0 and *dist* $>= 465.0$ **Remainder**: 0.8935382199962686 **Gain**: 0.024757614058220967
- *dist* < 470.0 and *dist* $>= 470.0$ **Remainder**: 0.9182958340544896 **Gain**: 0.0
- *dist* < 490.0 and *dist* $>= 490.0$ **Remainder**: 0.8455156082707569 **Gain**: 0.07278022578373267
- *dist* < 505.0 and *dist* $>= 505.0$ **Remainder**: 0.8999850522344305 **Gain**: 0.018310781820059074
- *dist* < 517.0 and *dist* $>= 517.0$ **Remainder**: 0.9182958340544896 **Gain**: 0.0
- *dist* < 520.0 and *dist* $>= 520.0$ **Remainder**: 0.8935382199962686 **Gain**: 0.024757614058220967
- *dist* < 530.0 and *dist* $>= 530.0$ **Remainder**: 0.8483857803777466 **Gain**: 0.06991005367674297

5.1.3 Speed

Best Gain is for *speed* < -100.0 and *speed* $>= -100.0$
Set 0: Kill (Yes): 0 Kill (No): 2
Set 1: Kill (Yes): 6 Kill (No): 1

Remainder: $\frac{2}{9} * B(\frac{0}{2}) + \frac{7}{9} * B(\frac{6}{7}) = 0.4601899388973658$

Remainder: 0.4601899388973658
Gain: 0.45810589515712374

The remaining split points and the corresponding gains are:

- *speed* < -120.0 and *speed* $>= -120.0$ **Remainder**: 0.7211361106303402 **Gain**: 0.19715972342414934
- *speed* < -100.0 and *speed* $>= -100.0$ **Remainder**: 0.4601899388973658 **Gain**: 0.45810589515712374
- *speed* < -90.0 and *speed* $>= -90.0$ **Remainder**: 0.7394468924503992 **Gain**: 0.17884894160409037
- *speed* < 117.0 and *speed* $>= 117.0$ **Remainder**: 0.8455156082707569 **Gain**: 0.07278022578373267
- *speed* < 120.0 and *speed* $>= 120.0$ **Remainder**: 0.8999850522344305 **Gain**: 0.018310781820059074
- *speed* < 120.0 and *speed* $>= 120.0$ **Remainder**: 0.8999850522344305 **Gain**: 0.018310781820059074
- *speed* < 147.0 and *speed* $>= 147.0$ **Remainder**: 0.8935382199962686 **Gain**: 0.024757614058220967
- *speed* < 220.0 and *speed* $>= 220.0$ **Remainder**: 0.7211361106303402 **Gain**: 0.19715972342414934

5.1.4 Attribute Choice

Clearly *speed* has the highest gain and is the most relevant. Specifically, the split *speed* < -100.0 and *speed* $>= -100.0$ is the optimum one.

Hence, this will be used at the root node of our decision tree.

This generates two children, one of whom have a common goal value - *kill* is No.

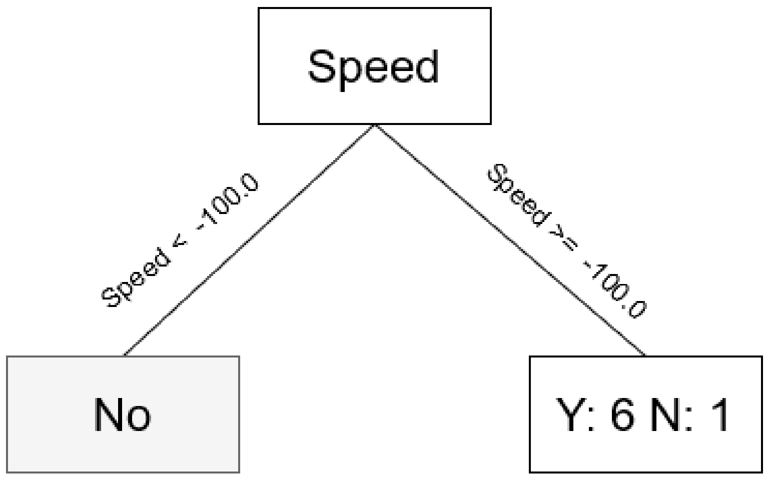

Figure 2: Tree at 1 level

5.2 Level 2

Since the previous layer had only one child without uniform truth value - we needn't worry about other nodes at the same level. We will now be concerned with the data set in Table 2 which represents the sub-tree to be formed.

Horizontal Angle(degree)	Distance(m)	Wind Speed(mph)	Kill
1.5	450	220	N
3	490	120	Y
5.5	530	117	Y
5.2	505	-90	Y
1.85	465	120	Y
4.8	517	147	Y
1.7	430	-100	Y

Table 2: New Data Set

The entropy of this set of examples is:

$$B(\frac{6}{7}) = \frac{6}{7}\log\frac{6}{7} + \frac{1}{7}\log\frac{1}{7} = 0.5916727785823275 \tag{5}$$

5.2.1 Angle

Best Gain is for $angle < 1.7$ and $angle >= 1.7$
Set 0: Kill (Yes): 0 Kill (No): 1
Set 1: Kill (Yes): 6 Kill (No): 0

Remainder: $\frac{1}{7} * B(\frac{0}{1}) + \frac{6}{7} * B(\frac{6}{6}) = 0.0$

Remainder: 0.0
Gain: 0.5916727785823275

The remaining split points and the corresponding gains are:

- *angle* < 1.7 and *angle* $>= 1.7$ **Remainder**: 0.0 **Gain**: 0.5916727785823275
- *angle* < 1.85 and *angle* $>= 1.85$ **Remainder**: 0.2857142857142857 **Gain**: 0.3059584928680418
- *angle* < 3.0 and *angle* $>= 3.0$ **Remainder**: 0.39355535745192405 **Gain**: 0.19811742113040343
- *angle* < 4.8 and *angle* $>= 4.8$ **Remainder**: 0.46358749969093305 **Gain**: 0.12808527889139443
- *angle* < 5.2 and *angle* $>= 5.2$ **Remainder**: 0.5156629249195446 **Gain**: 0.0760098536627829
- *angle* < 5.5 and *angle* $>= 5.5$ **Remainder**: 0.5571620756985892 **Gain**: 0.03451070288373825

5.2.2 Distance

Best Gain is for *dist* < 465.0 and *dist* $>= 465.0$

Set 0: Kill (Yes): 1 Kill (No): 1
Set 1: Kill (Yes): 5 Kill (No): 0

Remainder: $\frac{2}{7} * B(\frac{1}{2}) + \frac{5}{7} * B(\frac{5}{5}) = 0.2857142857142857$

Remainder: 0.2857142857142857
Gain: 0.3059584928680418

The remaining split points and the corresponding gains are:

- *dist* < 450.0 and *dist* $>= 450.0$ **Remainder**: 0.5571620756985892 **Gain**: 0.03451070288373825
- *dist* < 465.0 and *dist* $>= 465.0$ **Remainder**: 0.2857142857142857 **Gain**: 0.3059584928680418
- *dist* < 490.0 and *dist* $>= 490.0$ **Remainder**: 0.39355535745192405 **Gain**: 0.19811742113040343
- *dist* < 505.0 and *dist* $>= 505.0$ **Remainder**: 0.46358749969093305 **Gain**: 0.12808527889139443
- *dist* < 517.0 and *dist* $>= 517.0$ **Remainder**: 0.5156629249195446 **Gain**: 0.0760098536627829
- *dist* < 530.0 and *dist* $>= 530.0$ **Remainder**: 0.5571620756985892 **Gain**: 0.03451070288373825

5.2.3 Speed

Best Gain is for *speed* < 220.0 and *speed* $>= 220.0$

Set 0: Kill (Yes): 6 Kill (No): 0
Set 1: Kill (Yes): 0 Kill (No): 1

Remainder: $\frac{6}{7} * B(\frac{6}{6}) + \frac{1}{7} * B(\frac{0}{1}) = 0.0$

Remainder: 0.0
Gain: 0.5916727785823275

The remaining split points and the corresponding gains are:

- *speed* < -90.0 and *speed* $>= -90.0$ **Remainder**: 0.5571620756985892 **Gain**: 0.03451070288373825
- *speed* < 117.0 and *speed* $>= 117.0$ **Remainder**: 0.5156629249195446 **Gain**: 0.0760098536627829
- *speed* < 120.0 and *speed* $>= 120.0$ **Remainder**: 0.46358749969093305 **Gain**: 0.12808527889139443
- *speed* < 120.0 and *speed* $>= 120.0$ **Remainder**: 0.46358749969093305 **Gain**: 0.12808527889139443
- *speed* < 147.0 and *speed* $>= 147.0$ **Remainder**: 0.2857142857142857 **Gain**: 0.3059584928680418
- *speed* < 220.0 and *speed* $>= 220.0$ **Remainder**: 0.0 **Gain**: 0.5916727785823275

5.2.4 Attribute Choice

Clearly *angle* has the highest gain and is the most relevant. Specifically, the split *angle* < 1.7 and *angle* $>= 1.7$ is the optimum one.

Speed also perfectly divides the tree into two children with uniform truth values. We could choose *angle* or *speed* as a result - the gain value is the same.

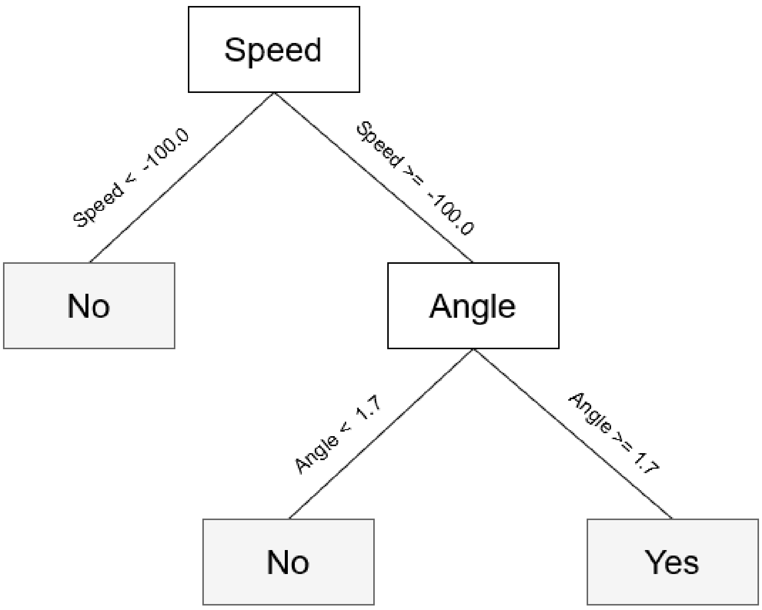

Figure 3: Tree with 2 Levels of tests