# Home Work 1 (CS6350)

Aishwarya Asesh (u1063384)

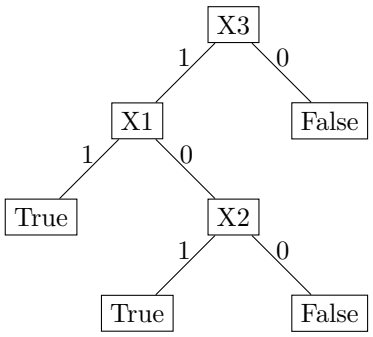
September 14, 2016

**1: Decision Trees**

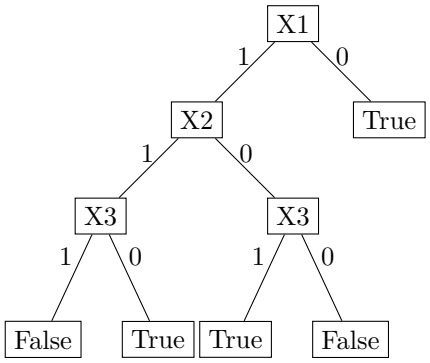
Question 1 : Represent the following Boolean functions as decision trees. (It is unnecessary to make the decision tree as small as possible; you can choose any root as you like. Use 1 for True and 0 for False. Also, note that an easy way to represent decision trees is as a series of if-then-else statements.)

Answer 1 :

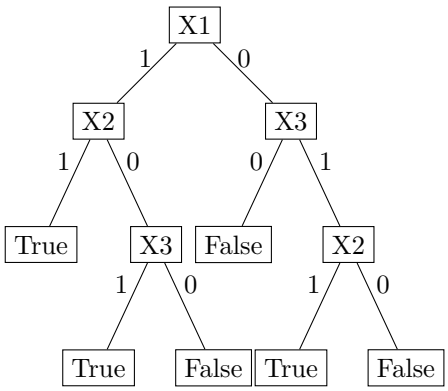
(a) Decision Tree for: $(x1 \vee x2) \wedge x3$




(b) Decision Tree for: $(x1 \wedge x2) xor (\neg x1 \vee x3)$



(c) Decision Tree for:
The 2-of-3 function defined as follows: at least 2 of $\{x1, x2, x3\}$ should be true for the output to be true.




Question 2 : When playing Pokemon Go, there is some chance that a Pokemon will be caught or it will escape. In the following question, build a decision tree to determine whether a Pokemon can be caught. There are four features:
(a) Berry (Yes or No) means whether a Razz Berry was used.
(b) Ball (Poke, Great, or Ultra) describes which kind of ball has been thrown.
(c) Color (Green, Yellow, or Red) stands for the diffculty level of catching this Pokemon.
(d) Type (Normal, Water, Flying, or Psychic) depicts the type of the Pokemon.

Answer 2 :

(a) Number of possible functions present there to map these four features to a Boolean decision $2^{16}$

(b) Entropy of the label = (-1) $((8/16)\log_2(8/16) + (8/16)\log_2(8/16)) = 1.0$

(c) Information Gain values for 4 attributes are as follows:

For attribute named "Berry"
"Yes" Entropy = (-1) $((6/7)\log_2(6/7) + (1/7)\log_2(1/7)) = 0.591$
"No" Entropy = (-1) $((2/9)\log_2(2/9) + (7/9)\log_2(7/9)) = 0.763$
Total = (7/16)*0.591 + (9/16)*0.763 = 0.687
Info Gain = 1 - 0.687

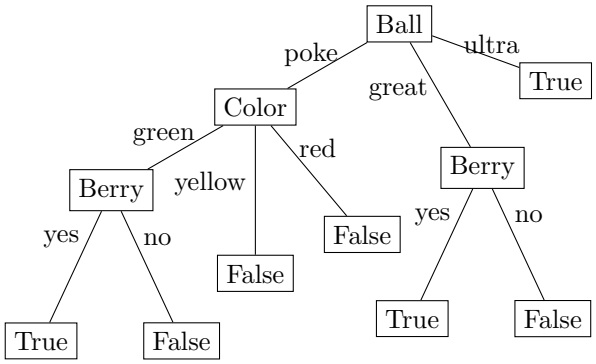
For attribute named "Ball"
"Poke" Entropy = (-1) $((1/6)\log_2(1/6) + (5/6)\log_2(5/6)) = 0.649$
"Great" Entropy = (-1) $((4/7)\log_2(4/7) + (3/7)\log_2(3/7)) = 0.984$
"Ultra" Entropy = (-1) $((3/3)\log_2(3/3)) = 0.0$
Total = (6/16)*0.649 + (7/16)*0.984 = 0.673
Info Gain = 1 - 0.673

For attribute named "Color"
"Green" Entropy = (-1) $((2/3)\log_2(2/3) + (1/3)\log_2(1/3)) = 0.917$
"Yellow" Entropy = (-1) $((3/7)\log_2(3/7) + (4/7)\log_2(4/7)) = 0.984$
"Red" Entropy = (-1) $((3/6)\log_2(3/6) + (3/6)\log_2(3/6)) = 1.0$
Total = (3/16)*0.917 + (7/16)*0.984 + (6/16)*1 = 0.976
Info Gain = 1 - 0.976

For attribute named "Type"
"Normal" Entropy = (-1) $((3/6)\log_2(3/6) + (3/6)\log_2(3/6)) = 1.0$
"Water" Entropy = (-1) $((2/4)\log_2(2/4) + (2/4)\log_2(2/4)) = 1.0$
"Flying" Entropy = (-1) $((3/4)\log_2(3/4) + (1/4)\log_2(1/4)) = 0.811$
"Psychic" Entropy = (-1) $((2/2)\log_2(2/2)) = 0.0$
Total = (6/16)*1.0 + (4/16)*1.0 + (4/16)*0.811 + 0 = 0.827
Info Gain = 1 - 0.827

(d) "Ball" should be considered the root node as it has highest Info Gain

(e) The tree constructed using the chosen root as "Ball" :




(f) Predicting the label in test data

Prediction for Test data number 1 is correct as the label turns out to be "YES"
Prediction for Test data number 2 is incorrect as the tree predicts it to be "YES", but it is mentioned as "NO"
Prediction for Test data number 3 is incorrect as the tree predicts it to be "YES", but it is mentioned as "NO"
Thus $Accuracy = (1/3) = 0.33\%$

(g) No, decision tree should not be used in the pokemon go case as it produces very little accuracy when used with the test data.

Question 3 : Recall that in the ID3 algorithm, we want to identify the best attribute that splits the examples that are relatively pure in one label. Apart from entropy, which you used in the previous question, there are other

methods to measure impurity. One such impurity measure is the Gini measure, that is used in the CART family of algorithms. Solve the previous problem using the Gini Index measure.

Answer 3 :

(a) Using the Gini index equation we can compute the values as follows:

Gini Index for label : (8/16)*(8/16)=0.25

Gini Index for "Berry" : (7/16)*(6/7 * 1/7) + (9/16)*(2/9 * 7/9) = 0.150
Info Gain = 1- 0.150

Gini Index for Ball: (6/16)*(1/6 * 5/6) + (7/16)*(4/7 * 3/7) = 0.159
Info Gain = 1- 0.159

Gini Index for Color: (3/16)*(2/3 * 1/3) + (7/16)*(3/7 * 4/7) + (6/16)*(1/2 * 1/2) = 0.242
Info Gain = 1- 0.242

Gini Index for Type: (6/16)*(1/2 * 1/2) + (4/16)*(1/2 * 1/2) + (4/16)*(3/4 * 1/4) = 0.203
Info Gain = 1- 0.203

(b) "Berry" is the root node according to Gini Index Measure as it has the highest Info Gain.
The two measures : Gini Index and ID3 doesn't lead to the same tree.

**2: Linear Classifiers**

Question 1 : Write a linear Classifier that correctly classifies the given dataset.

Answer 1 :
For the given dataset, we can define a linear classifier as:
$if\ x3 \vee x4$ then the value is True else False
According to the equation $b + w1x1 + w2x2 + ..... + wnxn = 0$
The linear classifier equation can be represented as $x3 + x4 - 0.5$
So the weights can be initialized as
$w1 = 0, w2 = 0, w3 = 1, w4 = 1$ and bias = -0.5

Question 2 : Suppose the dataset below is an extension of the above dataset. Check if your classifier from the previous quesion correctly classifies the given dataset. Report its accuracy.

Answer 2 :
According to the obtained classifier the data is correctly classified for 5 out of 7 cases.
The data is not classified correctly for the following cases:

When $x1 = 1, x2 = 1, x3 = 0, x4 = 0$ and when $x1 = 0x2 = 0x3 = 1x4 = 0$
Thus accuracy value is $5/7 = 0.714$ i.e. 71.4%

Question 3 : Write a linear classifier that correctly classifies the given dataset.

Answer 3 : A linear classifier that correctly classifies the whole dataset: We can observe the negative label values after combining all the three tables:

Table 1: Negation Values

| X1 | X2 | X3 | X4 | Label |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | -1 |
| 0 | 0 | 0 | 1 | -1 |
| 0 | 1 | 0 | 0 | -1 |
| 0 | 1 | 1 | 0 | -1 |

We observe that the values tend to be negative when $x1 = 0, x4 = 0$
According to the equation $b + w1x1 + w2x2 + ..... + wnxn = 0$
The generalized equation can be stated as $b + w1x1 + w4x4$ where
$w1 = 1, w2 = 0, w3 = 0, w4 = 1$ and bias value is 0.5
So the final equation can be stated as: $x1 + x4 - 0.5$

**3: Experiments**

**SETTING A**
1. Implementation
(a) The Program is Implemented using Python language.
For tree construction:
Total Data of training data file is stored in Total_data.
Features are stored in Features.
Class label column is stored in class_label_column.
Functions are created for calculating entropy and gain.
Recursive calls are made to the functions to give final values for tree construction.
(b) The error on SettingA/training.data file is reported as 0%.
Accuracy is 100% i.e. 1.0
(c) The error on SettingA/test.data file is reported as 0%.
Accuracy is 100% i.e. 1.0
(d) The Maximum depth of the decision tree is 3.

2. Limiting Depth
(a) The Table representation:

Table 2: Computation 2a

| Accuracy | Depth | Standard Deviation |
|---|---|---|
| 0.975928833072 | 1 | 0.0538247655494 |
| 0.975928833072 | 2 | 0.0538247655494 |
| 0.975928833072 | 3 | 0.0538247655494 |
| 0.975928833072 | 4 | 0.0538247655494 |
| 0.975928833072 | 5 | 0.0538247655494 |
| 0.975928833072 | 10 | 0.0538247655494 |
| 0.975928833072 | 15 | 0.0538247655494 |
| 0.975928833072 | 25 | 0.0538247655494 |

(b) 100% accuracy is reported.

**SETTING B**
1. Experiments
(a) The error on SettingB/training.data file is reported as 0%.
Accuracy is 100% i.e. 1.0
(b) The error on SettingB/test.data file is reported as 8.57%.
Accuracy is 91.4379% i.e. 0.9143
(c) The error on SettingA/training.data file is reported as 0.0006%
Accuracy is 99.9476% i.e. 0.9994
(d) The error on SettingA/test.data file is reported as 0.0017%
Accuracy is 99.8353% i.e. 0.9983
(e) The Maximum depth of the decision tree is 9.

2. Limiting Depth
(a) Highest accuracy is reported for 1. So it is the best depth.

Table 3: Computation 2a

| Accuracy | Depth | Standard Deviation |
|---|---|---|
| 0.901883830455 | 1 | 0.0958846175832 |
| 0.919937205651 | 2 | 0.0430400942229 |
| 0.911826268969 | 3 | 0.0352830378007 |
| 0.90083725798 | 4 | 0.0431774569973 |
| 0.897959183673 | 5 | 0.0414454592855 |
| 0.895866038723 | 10 | 0.0426245470061 |
| 0.895866038723 | 15 | 0.0426245470061 |
| 0.895866038723 | 25 | 0.0426245470061 |

(b) 93.85% accuracy is reported.

**SETTING C**

1. To handle the missing value using Method 1, the function searched for missing attribute in every row, and returned its column number. The most frequent values for the columns were then calculate using the function find unique. The missing attribute was then replaced with most frequent charachter. For Method 2, column of the missing attribute was found as in the previous. A new function calculated the most frequent attribute value and missing attribute was replaced with that class label attribute. However, due to the last minute errors, codes for these changes are not submitted.

2.

Table 4: Computation 2a

| Accuracy | Depth | Standard Deviation |
|---|---|---|
| 0.978860347492 | 1 | 0.0384691797872 |
| 0.979963695505 | 2 | 0.037042433439 |
| 0.98309445057 | 3 | 0.0378019577233 |
| 0.98309445057 | 4 | 0.0378019577233 |
| 0.98309445057 | 5 | 0.0378019577233 |
| 0.98309445057 | 10 | 0.0378019577233 |
| 0.98309445057 | 15 | 0.0378019577233 |
| 0.98309445057 | 25 | 0.0378019577233 |

3. Accuracy of 100% was acheived using the method 3 used above.