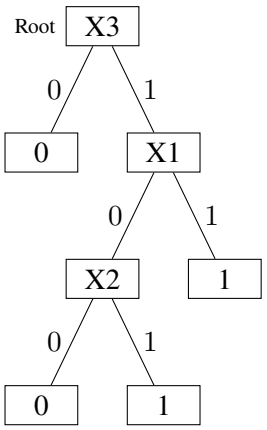
# CS 6350: Machine Learning – Homework 1

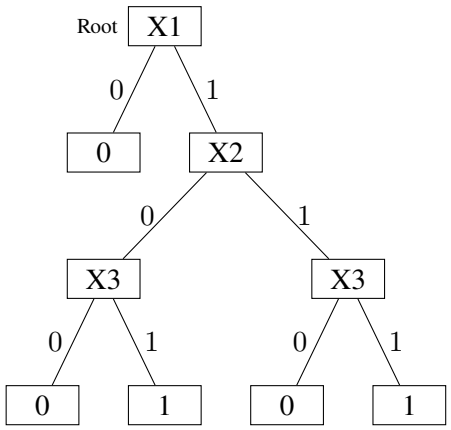
Clinton Fernandes, u1016390 09/13/2016

## 1 Decision trees

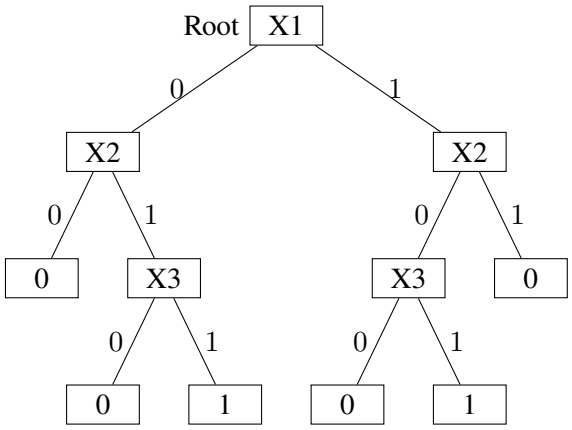
1. **Representing boolean functions as decision trees**




*tree: 1.a*




*tree: 1.b*




*tree: 1.c*

2. **Decision tree to check if Pokemon is caught**

   (a) Possible function $= 2^{(2*3*3*4)} = 2^{(72)} = 4.7223665e + 21$

   (b) p = 8/16 = 0.5 n = 8/16 = 0.5

   Entropy(Caught) $= -(p * log_2(p) + n * log_2(n))$

   Entropy(Caught) $= -(0.5 * log_2(0.5) + 0.5 * log_2(0.5)) = 1$

(c) Berry = Yes : 7/16 examples

p = 6/7 n = 1/7 $H_{yes} = 0.592$

Berry = No: 9/16 examples

p = 6/7 n = 1/7 $H_{yes} = 0.764$

**Expected entropy** : $= (7/16) * H_{yes} + (9/16) * H_{no} = 0.689$

**Information Gain (Berry)** : $1 - 0.689 - = 0.311$

Ball = Poke : 6/16 examples

p = 1/6 n = 5/6 $H_{poke} = 0.65$

Ball = Great : 7/16 examples

p = 4/7 n = 3/7 $H_{great} = 0.985$

Ball = Ultra : 3/16 examples

p = 3/3 n = 0/3 $H_{ultra} = 0.0$

**Expected entropy** : $= (6/16) * H_{poke} + (7/16) * H_{great} + (3/16) * H_{ultra} = 0.675$

**Information Gain (Ball)** : $1 - 0.675 = 0.325$

Color = Green : 3/16 examples

p = 2/3 n = 1/3 $H_{green} = 0.918$

Color = Yellow : 7/16 examples

p = 3/7 n = 4/7 $H_{yellow} = 0.985$

Color = Red : 6/16 examples

p = 3/6 n = 3/6 $H_{red} = 1$

**Expected entropy** : $= (3/16) * H_{green} + (7/16) * H_{yellow} + (6/16) * H_{red} = 0.978$

**Information Gain (Color)** : $1 - 0.978 = 0.022$

Type = Normal : 6/16 examples p = 3/6 n = 3/6 $H_{normal} = 1$

Type = Water : 4/16 examples p = 2/4 n = 2/4 $H_{water} = 1$

Type = Flying : 4/16 examples p = 3/4 n = 1/4 $H_{flying} = 0.811$

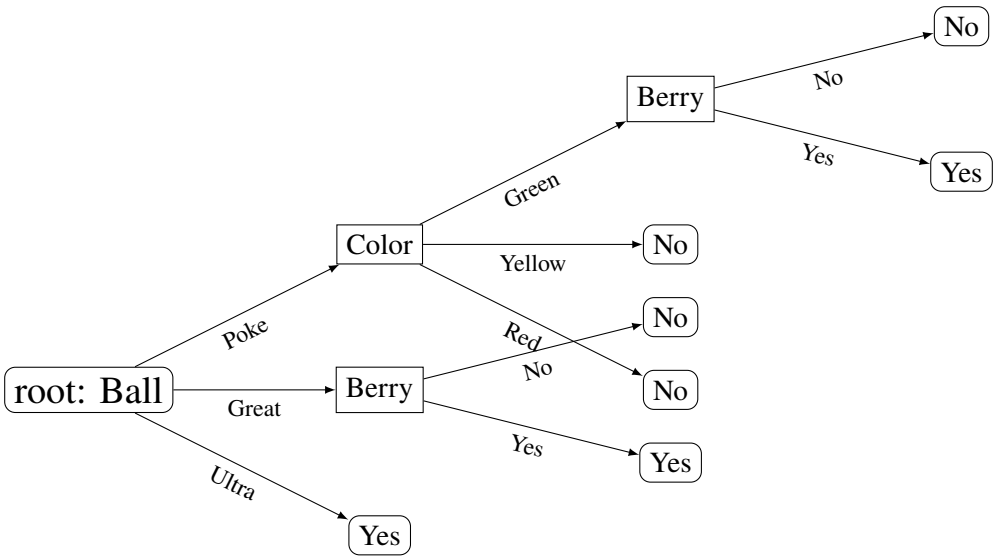
Type = Psychic : 2/16 examples p = 0/2 n = 2/2 $H_{psychic} = 0.0$

**Expected entropy** : $= (6/16) * H_{normal} + (4/16) * H_{water} + (4/16) * H_{flying} + (2/16) * H_{psychic} = 0.828$

**Information Gain (Type)** : $1 - 0.978 = 0.172$

(d) The attribute 'Ball' should be the root for the decision tree as it has the most Information gain as compared to the other attributes.

(e) Decision tree:




(f)

| Test | Predicted |
|---|---|
| Yes | Yes |
| No | Yes |
| No | Yes |

- The table on the left shows the label (Caught) for both the actual test data and predicted label. It was seen that the prediction was incorrect for 2 out of 3 times. The accuracy is 0.333

(g) From the previous question, it was seen that the accuracy was poor (1/3=0.33). Hence decision tree alone is not a good idea for the Pokemon problem.

3. **Using Gini coefficient instead of Entropy**

(a) Berry = Yes : 7/16 examples

p = 6/7 n = 1/7 $G_{yes} = 0.245$

Berry = No: 9/16 examples

p = 6/7 n = 1/7 $G_{yes} = 0.346$

**Expected Gini coeff** : $= (7/16) * G_{yes} + (9/16) * G_{no} = 0.302$

**Information Gain (Berry)** : $0.5 - 0.302- = 0.198$

Ball = Poke : 6/16 examples

p = 1/6 n = 5/6 $G_{poke} = 0.278$

Ball = Great : 7/16 examples

p = 4/7 n = 3/7 $G_{great} = 0.49$

Ball = Ultra : 3/16 examples

p = 3/3 n = 0/3 $G_{ultra} = 0.0$

**Expected Gini coeff** : $= (6/16) * G_{poke} + (7/16) * G_{great} + (3/16) * G_{ultra} = 0.318$

**Information Gain (Ball)** : $0.5 - 0.318 = 0.182$

Color = Green : 3/16 examples

p = 2/3 n = 1/3 $G_{green} = 0.444$

Color = Yellow : 7/16 examples

p = 3/7 n = 4/7 $G_{yellow} = 0.49$

Color = Red : 6/16 examples

p = 3/6 n = 3/6 $G_{red} = 0.5$

**Expected Gini coeff** : $= (3/16) * G_{green} + (7/16) * G_{yellow} + (6/16) * G_{red} = 0.485$

**Information Gain (Color)** : $0.5 - 0.485 = 0.015$

Type = Normal : 6/16 examples p = 3/6 n = 3/6 $G_{normal} = 0.5$

Type = Water : 4/16 examples p = 2/4 n = 2/4 $G_{water} = 0.5$

Type = Flying : 4/16 examples p = 3/4 n = 1/4 $G_{flying} = 0.375$

Type = Psychic : 2/16 examples p = 0/2 n = 2/2 $G_{psychic} = 0.0$

**Expected Gini coeff** : $= (6/16) * G_{normal} + (4/16) * G_{water} + (4/16) * G_{flying} + (2/16) * G_{psychic} = 0.406$

**Information Gain (Type)** : $0.5 - 0.406 = 0.094$

(b) From the result of the previous question, the attribute with the highest Information gain is 'Berry'. Berry is the root for the decision tree.
The two measures do not lead to the same tree.

## 2 Linear Classifiers

1. From the given logic table, it appears that the label is '1' if all the four features are '1' and label is '-1' if all the four features are '0'. We can use the following disjunction expression to get the required label:
$y = x_1 \cup x_2 \cup x_3 \cup x_4$ This is equivalent to: $x_1 + x_2 + x_3 + x_4 \geqslant 1$
Here, the weight is: $w^T = [1\,1\,1\,1]$ and bias is: $b = -1$. Also the feature space $x^T = [x_1\ x_2\ x_3\ x_4]$
Such that, if $w^T x + b \geqslant 0$ then $y = 1$
if $w^T x + b \leqslant 0$ then $y = -1$

2. Using the classifier from the previous question ' $y = x_1 \cup x_2 \cup x_3 \cup x_4$', we get the results as shown in table 2.2.
It was seen that the accuracy was 9/10 or 90%

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $o$ | predict | match |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | yes |
| 0 | 0 | 1 | 1 | 1 | 1 | yes |
| 0 | 0 | 0 | 0 | -1 | -1 | yes |
| 1 | 0 | 1 | 1 | 1 | 1 | yes |
| 0 | 1 | 0 | 1 | 1 | 1 | yes |
| 1 | 0 | 1 | 0 | 1 | 1 | yes |
| 1 | 1 | 0 | 0 | 1 | 1 | yes |
| 1 | 1 | 1 | 1 | 1 | 1 | yes |
| 1 | 1 | 1 | 0 | 1 | 1 | yes |
| 0 | 0 | 1 | 0 | -1 | 1 | no |

*table 2.2*

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $o$ | predict | match |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | yes |
| 0 | 0 | 1 | 1 | 1 | 1 | yes |
| 0 | 0 | 0 | 0 | -1 | -1 | yes |
| 1 | 0 | 1 | 1 | 1 | 1 | yes |
| 0 | 1 | 0 | 1 | 1 | 1 | yes |
| 1 | 0 | 1 | 0 | 1 | 1 | yes |
| 1 | 1 | 0 | 0 | 1 | 1 | yes |
| 1 | 1 | 1 | 1 | 1 | 1 | yes |
| 1 | 1 | 1 | 0 | 1 | 1 | yes |
| 0 | 0 | 1 | 0 | -1 | -1 | yes |
| 0 | 1 | 0 | 0 | -1 | -1 | yes |
| 0 | 1 | 1 | 0 | -1 | -1 | yes |
| 0 | 1 | 1 | 1 | 1 | 1 | yes |
| 1 | 0 | 0 | 0 | 1 | 1 | yes |
| 1 | 0 | 0 | 1 | 1 | 1 | yes |
| 1 | 1 | 0 | 1 | 1 | 1 | yes |

*table 2.3*

3. After looking at the logic table, and noticing how $x_1, x_2, x_3$ and $x_4$ affect the output, It was seen that the output was 1 when either $x_1$ or $x_4$ was 1. The output was -1 when both $x_1$ and $x_4$ were 0.
   We can use the following disjunction expression to get the required label: $y = x_1 \cup x_4$
   The following expression with linear separators can be used as a linear classifier: $x_1 + x_4 \geqslant 1$
   Here, the weight is: $w^T = [1\,0\,0\,1]$ and bias is: $b = -1$. Also the feature space $x^T = [x_1\ x_2\ x_3\ x_4]$
   Such that, if $w^T x + b \geqslant 0$ then $y = 1$ , and if $w^T x + b \leqslant 0$ then $y = -1$
   The result, using this classifier on the whole dataset is given in table 2.3. The accuracy is 100%

# 3 Experiments

## Setting A

1. **Implementation**

   (a) The decision tree data structure along with the ID3 algorithm was implemented using Python programming language.
   The dictionary data structure was used in order to store the nodes and the branches of the tree.
   After reading the data from 'datasets folder', it was stored as list of lists.

(b) The error of the decision tree on the `SettingA/training.data` file was found to the 0%. The accuracy was 100%.

(c) The error of the decision tree on the `SettingA/test.data` file was 0%, accuracy was 100%.

(d) The maximum depth of the tree was 3.

2. **Limiting Depth**

(a)

| depth | % accuracy | std. dev on % accuracy |
|---|---|---|
| 1 | 97.65 | 5.126 |
| 2 | 98.14 | 4.154 |
| 3 | 98.14 | 4.154 |
| 4 | 98.14 | 4.154 |
| 5 | 98.14 | 4.154 |
| 10 | 98.14 | 4.154 |
| 15 | 98.14 | 4.154 |
| 20 | 98.14 | 4.154 |

(b) It was found that the depths 2,3,4,5,10,15,20 gave the same average cross-validation accuracy which was highest. When the tree was trained with depth = 2, the accuracy on the test data was 99.78%. But when the tree was trained with depth = 3, the accuracy on the test data was 100%.

## Setting B

1. **Experiments**

(a) Error of decision tree on the `SettingB/training.data` is 0.000%

(b) Error of decision tree on the `SettingB/test.data` file is 6.476%

(c) Error of decision tree on the `SettingA/training.data` file is 0.052%

(d) Error of decision tree on the `SettingA/test.data` file is 0.220%

(e) Maximum depth of decision tree is 9

2. **Limiting Depth**

(a)

| depth | % accuracy | std. dev on % acc |
|---|---|---|
| 1 | 60.596 | 31.596 |
| 2 | 92.883 | 4.297 |
| 3 | 90.215 | 9.817 |
| 4 | 93.249 | 2.156 |
| 5 | 92.49 | 2.839 |
| 10 | 92.229 | 2.897 |
| 15 | 92.229 | 2.897 |
| 20 | 92.229 | 2.897 |

- The cross-validation accuracy and standard deviation for each depth are sh below in the table. A depth of 4 should be chosen as the best because it gives the highest accuracy of 93.249%

(b) The accuracy of the decision tree on the `SettingB/test.data` file was 94.182%

## Setting C

1. The decision tree implementation was updated so that different methods to deal with missing features could be used.

   For Method 1: The missing feature was replaced by a feature value that appeared the most number of times within the attribute.

   Method 2: In this, the missing feature was replaced with the most common value of that attribute. The most common value was chosen from amongst values of the attribute that had the same label as the missing attribute.

   Method 3: In this, the missing feature was treated as a special feature; such that it was considered to be present like any other feature.

2. The table below gives the accuracy and standard deviation fo each method.

   notations in table:

   m1 = method 1 m2 = method 2 m3 = method 3

   acc = % accuracy std.dev = standard deviation on % accuracy

| depth | acc(m1) | std.dev(m1) | acc(m2) | std.dev(m2) | acc(m3) | std.dev(m3) |
|---|---|---|---|---|---|---|
| 1 | 73.557 | 25.674 | 73.557 | 25.674 | 73.557 | 25.674 |
| 2 | 97.886 | 3.846 | 97.886 | 3.846 | 97.886 | 3.846 |
| 3 | 98.823 | 1.82 | 98.823 | 1.82 | 98.823 | 1.82 |
| 4 | 99.173 | 1.849 | 99.173 | 1.849 | 99.173 | 1.849 |
| 5 | 99.173 | 1.849 | 99.173 | 1.849 | 99.173 | 1.849 |
| 10 | 99.173 | 1.849 | 99.173 | 1.849 | 99.173 | 1.849 |
| 15 | 99.173 | 1.849 | 99.173 | 1.849 | 99.173 | 1.849 |
| 20 | 99.173 | 1.849 | 99.173 | 1.849 | 99.173 | 1.849 |

3. From the the result of the previous question (table above), it can be seen that all three methods reult in the same accuracy, and hence any method can be chosen. I used the $3^{rd}$ method.

   It was found that the accuracy of the tree on `SettingC/test.data` = 100%.