Decision Trees

\[ Y = y \]

Input Data

\[ X_1 = x_1 \]

\[ \vdots \]

\[ X_M = x_M \]

Classifier

Class prediction

\[ Y = y \]

Training data
Decision Tree Example

- Three variables:
  - Hair = {blond, dark}
  - Height = {tall, short}
  - Country = {Gromland, Polvia}

Training data:
(B,T,P)  P:2 G:4
(B,T,P)  Hair = B?
(B,S,G)  Hair = D?
(D,S,G)  P:2 G:2
(D,T,G)  Height = T?
(B,S,G)  Height = S?
         P:2 G:0
         P:0 G:2

At each level of the tree, we split the data according to the value of one of the attributes. After enough splits, only one class is represented in the node. This is a terminal leaf of the tree. We call that class the output class for that node. ‘G’ is the output for this node.
The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

(B, T) is classified as P

(D, S) is classified as G

---

**General Case (Discrete Attributes)**

- We have $R$ observations from training data
  - Each observation has $M$ attributes $X_1, \ldots, X_M$
  - Each $X_i$ can take $N$ distinct discrete values
  - Each observation has a class attribute $Y$ with $C$ distinct (discrete) values
  - Problem: Construct a sequence of tests on the attributes such that, given a new input $(x_1, \ldots, x_M)$, the class attribute $y$ is correctly predicted

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$\ldots$</th>
<th>$X_M$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>$x_1'$</td>
<td>$\ldots$</td>
<td>$x_M'$</td>
<td>???</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$X_1$</th>
<th>$\ldots$</th>
<th>$X_M$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data 1</td>
<td>$x_1$</td>
<td>$\ldots$</td>
<td>$x_M$</td>
<td>$y$</td>
</tr>
<tr>
<td>Data 2</td>
<td>$\ldots$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data $R$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Training Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$X =$ attributes of training data ($RxM$)  $Y =$ Class of training data ($R$)
General Decision Tree (Discrete Attributes)

\[ X_1 = \text{first possible value for } X_1 ? \]

\[ X_1 = \text{nth possible value for } X_1 ? \]

\[ X_j = \text{ith possible value for } X_j ? \]

Output class \( Y = y_1 \)

Decision Tree Example

\[ X_2 = 0.5 \]

\[ X_1 = 0.5 \]

\[ X_1 < 0.5 ?? \]

\[ X_2 < 0.5 ?? \]
The class of a new input can be classified by following the tree all the way down to a leaf and by reporting the output of the leaf. For example:

(0.2,0.8) is classified as

(0.8,0.2) is classified as

General Case (Continuous Attributes)

- We have \( R \) observations from training data
  - Each observation has \( M \) attributes \( X_1, \ldots, X_M \)
  - Each \( X_i \) can take \( N \) distinct discrete values
  - Each observation has a class attribute \( Y \) with \( C \) distinct (discrete) values
  - Problem: Construct a sequence of tests of the form \( X_i < t_i \) on the attributes such that, given a new input \( (x_1, \ldots, x_M) \), the class attribute \( y \) is correctly predicted

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Input data} & x_1 & \ldots & x_M & Y \\
\hline
\text{Data 1} & x_1 & \ldots & x_M & Y \\
\hline
\text{Data 2} & & & & \\
\hline
\vdots & & & & \\
\hline
\text{Data } R & & & & \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
X_1 & \ldots & X_M & Y \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
X_1 & \ldots & X_M & Y \\
\hline
\end{array}
\]

\[X = \text{attributes of training data} \ (R \times M) \quad Y = \text{Class of training data} \ (R)\]
General Decision Tree (Continuous Attributes)

\[ X_t < t_1 ? \]

Output class \( Y = y_1 \)

\[ X_j < t_j ? \]

Output class \( Y = y_c \)

Basic Questions

- How to choose the attribute/value to split on at each level of the tree?
- When to stop splitting? When should a node be declared a leaf?
- If a leaf node is impure, how should the class label be assigned?
- If the tree is too large, how can it be pruned?
How to choose the attribute/value to split on at each level of the tree?

- Two classes (red circles/green crosses)
- Two attributes: $X_1$ and $X_2$
- 11 points in training data
- Idea → Construct a decision tree such that the leaf nodes predict correctly the class for all the training examples
This node is “pure” because there is only one class left \( \rightarrow \) No ambiguity in the class label.

This node is almost “pure” \( \rightarrow \) Little ambiguity in the class label.

These nodes contain a mixture of classes \( \rightarrow \) Do not disambiguate between the classes.

We want to find the most compact, smallest size tree (Occam’s razor), that classifies the training data correctly \( \rightarrow \) We want to find the split choices that will get us the fastest to pure nodes.

This node is “pure” because there is only one class left \( \rightarrow \) No ambiguity in the class label.

This node is almost “pure” \( \rightarrow \) Little ambiguity in the class label.

These nodes contain a mixture of classes \( \rightarrow \) Do not disambiguate between the classes.
Digression: Information Content

- Suppose that we are dealing with data which can come from four possible values (A, B, C, D)
- Each class may appear with some probability
- Suppose \( P(A) = P(B) = P(C) = P(D) = 1/4 \)
- What is the average number of bits necessary to encode each class?
- In this case: average = \( 2 = 2 \cdot P(A) + 2 \cdot P(B) + 2 \cdot P(C) + 2 \cdot P(D) \)
  - A \( \rightarrow \) 00 B \( \rightarrow \) 01 C \( \rightarrow \) 10 D \( \rightarrow \) 11

The distribution is not very informative \( \rightarrow \) impure

Information Content

- Suppose now \( P(A) = 1/2 \) \( P(B) = 1/4 \) \( P(C) = 1/8 \) \( P(D) = 1/8 \)
- What is the average number of bits necessary to encode each class?
- In this case, the classes can be encoded by using 1.75 bits on average
- \( A \rightarrow 0 \) B \( \rightarrow 10 \) C \( \rightarrow 110 \) D \( \rightarrow 111 \)
- Average
  \[
  = 1 \cdot P(A) + 2 \cdot P(B) + 3 \cdot P(C) + 3 \cdot P(D) = 1.75
  \]

The distribution is more informative \( \rightarrow \) higher purity
Entropy

• In general, the average number of bits necessary to encode \( n \) values is the entropy:

\[
H = -\sum_{i=1}^{n} P_i \log_2 P_i
\]

• \( P_i \) = probability of occurrence of value \( i \)
  – High entropy \( \rightarrow \) All the classes are (nearly) equally likely
  – Low entropy \( \rightarrow \) A few classes are likely; most of the classes are rarely observed

The entropy captures the degree of “purity” of the distribution
Example Entropy Calculation

1

\[ N_A = 1 \]
\[ N_B = 6 \]
\[ p_A = \frac{N_A}{N_A + N_B} = \frac{1}{7} \]
\[ p_B = \frac{N_B}{N_A + N_B} = \frac{6}{7} \]

\[ H_1 = -p_A \log_2 p_A - p_B \log_2 p_B \]
\[ = 0.59 \]

2

\[ N_A = 3 \]
\[ N_B = 2 \]
\[ p_A = \frac{N_A}{N_A + N_B} = \frac{3}{5} \]
\[ p_B = \frac{N_B}{N_A + N_B} = \frac{2}{5} \]

\[ H_2 = -p_A \log_2 p_A - p_B \log_2 p_B \]
\[ = 0.97 \]

\[ H_1 < H_2 \Rightarrow (2) \text{ less pure than } (1) \]
Conditional Entropy

Entropy before splitting: $H$

After splitting, a fraction $P_L$ of the data goes to the left node, which has entropy $H_L$

After splitting, a fraction $P_R$ of the data goes to the left node, which has entropy $H_R$

The average entropy after splitting is:

$$H_L \times P_L + H_R \times P_R$$
Information Gain

We want nodes as pure as possible
→ We want to reduce the entropy as much as possible
→ We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

Maximize:

\[ IG = H - (H_L \times P_L + H_R \times P_R) \]

Notations

- Entropy: \( H(Y) \) = Entropy of the distribution of classes at a node
- Conditional Entropy:
  - Discrete: \( H(Y | X_j) \) = Entropy after splitting with respect to variable \( j \)
  - Continuous: \( H(Y | X_j, t) \) = Entropy after splitting with respect to variable \( j \) with threshold \( t \)
- Information gain:
  - Discrete: \( IG(Y | X_j) = H(Y) - H(Y | X_j) \) = Entropy after splitting with respect to variable \( j \)
  - Continuous: \( IG(Y | X_j, t) = H(Y) - H(Y | X_j, t) \) = Entropy after splitting with respect to variable \( j \) with threshold \( t \)
Information Gain

We want nodes as pure as possible
→ We want to reduce the entropy as much as possible
→ We want to maximize the difference between the entropy of the parent node and the expected entropy of the children

Maximize:

Information Gain (IG) = Amount by which the ambiguity is decreased by splitting the node

\[ IG = H - (H_L \times P_L + H_R \times P_R) \]

\[ IG = H - (H_L \times \frac{4}{11} + H_R \times \frac{7}{11}) \]

\[ IG = H - (H_L \times \frac{5}{11} + H_R \times \frac{6}{11}) \]
Choose this split because the information gain is greater than with the other split.
More Complete Example

= 20 training examples from class A
= 20 training examples from class B
Attributes = $X_1$ and $X_2$ coordinates

Best split value (max Information Gain) for $X_1$ attribute: 0.24 with IG = 0.138
Best split value (max Information Gain) for $X_2$ attribute: 0.234 with IG = 0.202

Best $X_1$ split: 0.24, IG = 0.138
Best $X_2$ split: 0.234, IG = 0.202

Split on $X_2$ with 0.234

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B
Best $X$ split: 0.24, IG = 0.138

There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’

Total data points = 7
7 A
0 B

Total data points = 33
13 A
20 B

Best $X_1$ split value:
$X_1 \leq 0.233$

Best split value (max Information Gain) for $X_1$ attribute: 0.22 with IG ~ 0.182
Best split value (max Information Gain) for $X_2$ attribute: 0.75 with IG $\sim$ 0.353

Best $X_1$ split: 0.22, IG = 0.182
Best $X_2$ split: 0.75, IG = 0.353

Split on $X_2$ with 0.75
There is no point in splitting this node further since it contains only data from a single class → return it as a leaf node with output ‘A’

Each of the leaf nodes is pure → contains data from only one class
Final decision tree
Given an input \((X,Y)\) →
Follow the tree down to a leaf.
Return corresponding output class for this leaf

Example \((X,Y) = (0.5,0.5)\)

Basic Questions

• How to choose the attribute/value to split on at each level of the tree?

• When to stop splitting? When should a node be declared a leaf?

• If a leaf node is impure, how should the class label be assigned?

• If the tree is too large, how can it be pruned?
Pure and Impure Leaves and When to Stop Splitting

All the data in the node comes from a single class → We declare the node to be a leaf and stop splitting. This leaf will output the class of the data it contains.

Several data points have exactly the same attributes even though they are from the same class → We cannot split any further → We still declare the node to be a leaf, but it will output the class that is the majority of the classes in the node (in this example, ‘B’).

Decision Tree Algorithm (Continuous Attributes)

- LearnTree$(X, Y)$
  - Input:
    - Set $X$ of $R$ training vectors, each containing the values $(x_1, ..., x_M)$ of $M$ attributes $(X_1, ..., X_M)$
    - A vector $Y$ of $R$ elements, where $y_j$ = class of the $j$th datapoint
  - If all the datapoints in $X$ have the same class value $y$
    - Return a leaf node that predicts $y$ as output
  - If all the datapoints in $X$ have the same attribute value $(x_{j_1}, ..., x_{j_M})$
    - Return a leaf node that predicts the majority of the class values in $Y$ as output
  - Try all the possible attributes $X_j$ and threshold $t$ and choose the one, $j^*$, for which IG$(Y|X_j, t)$ is maximum
  - $X_{j^*}Y_{j^*}$ = set of datapoints for which $x_j < t$ and corresponding classes
  - $X_{j^*}Y_{j^*}$ = set of datapoints for which $x_j \geq t$ and corresponding classes
  - Left Child $\leftarrow$ LearnTree$(X_{j^*}, Y_{j^*})$
  - Right Child $\leftarrow$ LearnTree$(X_{j^*}, Y_{j^*})$
Decision Tree Algorithm (Discrete Attributes)

- **LearnTree**(X,Y)
  - Input:
    - Set X of R training vectors, each containing the values (x₁,..,xₘ) of M attributes (X₁,..,Xₘ)
    - A vector Y of R elements, where yⱼ = class of the jth datapoint
  - If all the datapoints in X have the same class value y
    - Return a leaf node that predicts y as output
  - If all the datapoints in X have the same attribute value (x₁,..,xₘ)
    - Return a leaf node that predicts the majority of the class values in Y as output
  - Try all the possible attributes Xᵢ and choose the one, j*, for which IG(Y|Xⱼ) is maximum
  - For every possible value v of Xⱼ:
    - Xᵥ, Yᵥ = set of datapoints for which xⱼ = v and corresponding classes
    - Childᵥ ← LearnTree(Xᵥ, Yᵥ)

---

Decision Trees So Far

- Given R observations from training data, each with M attributes X and a class attribute Y, construct a sequence of tests (decision tree) to predict the class attribute Y from the attributes X
- Basic strategy for defining the tests (“when to split”) → maximize the information gain on the training data set at each node of the tree
- Problems (next):
  - Computational issues → How expensive is it to compute the IG
  - The tree will end up being much too big → pruning
  - Evaluating the tree on training data is dangerous → overfitting