Producing a Decision Tree From a Training Set

- Several algorithms have been developed for constructing a decision tree from a training set
  - We discuss the **ID3 algorithm** (top-down)

- ID3 starts by selecting the attribute to be used at the top level of the tree to make the first decision

- This decision yields the nodes at the second level of the tree. The procedure repeats for each of these nodes, and so on.
Picking the Top-Most Attribute

- Intuitively we want to pick the attribute that gives the "most information" about the final decision

- The ID3 algorithm uses the **entropy** measure from Information Theory

  \[ \text{entropy(TrainingTable)} = - \sum_{i \in \text{outcomes}} p_i \log_2 p_i \]

  \( p_i \) = probability of the outcome \( i \) in TrainingTable

- Practically: \( p_i \) is approximated as

  \[ p_i = \frac{\# \text{items in the table with outcome}=i}{\# \text{of all items in the table}} \]

Properties of the Entropy – \( \Sigma p_i \log_2 p_i \)

- Entropy determines the degree of randomness in the data:
  - \( p_{\text{yes}} = p_{\text{no}} = \frac{1}{2} \) – data is completely random
    \[ \text{entropy} = - \frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = \frac{1}{2} + \frac{1}{2} = 1 \]
  - \( p_{\text{yes}} = 1, p_{\text{no}} = 0 \) or \( p_{\text{no}} = 1, p_{\text{yes}} = 0 \) – data is totally nonrandom
    \[ \text{entropy} = - 1 \log_2 1 - 0 \log_2 0 = 0 \]

- The lower the entropy – the less randomness exists in the data \( \equiv \) the more information exists in that data
Information Gain

- For the entire table, 6 entries have the outcome “Yes” and 14 have the outcome “No”
  - So the entropy of the entire table is
    \[- \frac{6}{20} \log_2 \frac{6}{20} + \frac{14}{20} \log_2 \frac{14}{20} \approx .881\]

- The ID3 algorithm selects as the top-most node the attribute that provides the largest information gain (explained next)

Information Gain (cont’d)

- Select an attribute, A, and compute the entropies of the subtrees w.r.t. that attribute

- Information gain for A:
  \[ \text{entropy} \rightarrow (\Sigma_{i=1...n} \text{entropy}_i \times \text{weight}_i) \]
  - \[ \text{weight} = \frac{\text{tuples}(T_i)}{\text{tuples}(T)} \]
  - How much less random the data has become after splitting the training set into subtrees
Information Gain (con’t)

- If the top-most node in the tree were Previous Default, there would be two subtrees:
  - a subtree with Previous Default = “Yes”
  - a subtree with Previous Default = “No”

- The entropies of these two subtrees would be
  - For Previous Default = “Yes”:
    - 4 of the 6 entries have the outcome “Yes” and 2 have “No”
      - The entropy is \(-\frac{4}{6} \log_2 \frac{4}{6} – \frac{2}{6} \log_2 \frac{2}{6} = .918\)
  - For Previous Default = “No”:
    - 2 of the 14 entries have the outcome “Yes” and 12 have “No”
      - The entropy is \(-\frac{2}{14} \log_2 \frac{2}{14} – \frac{12}{14} \log_2 \frac{12}{14} = .592\)

- The average entropy of these subtrees is

\[
\frac{6}{20} \times 0.918 + \frac{14}{20} \times 0.592 = 0.690
\]

- The Information Gain from using Previous Default as the top attribute is

\[
0.881 – 0.690 = 0.191
\]
Comparing Information Gains

- **Previous Default** as the top-most attribute
  - The information gain = .191
- **Married** as the top-most attribute
  - The information gain = .056
- **Income** as the top-most attribute
  - Must compute information gain for all possible ranges
  - For example for the range Income < 50 and Income >= 50 the Information Gain is .031 --- discretization

- The maximum Information Gain turns out to be for the attribute **Previous Default**, so we select that as the top-most attribute in the decision tree

The Rest of the Tree

- Repeat the process on each of the subtrees
  - Different subtrees might have different top-most nodes and/or different ranges for **Income**
  - If all nodes in a subtree have the same outcome:
    - the subtree becomes a leaf node and the procedure stops for that subtree
  - If not all nodes in a subtree have the same outcome:
    - If there are no more attributes to use: That subtree becomes a leaf node and the procedure stops for that subtree
      - The classification rules that access such a subtree will incorrectly classify some data.
      - E.g., the subtree PreviousDefault = No AND Married = Yes AND Income >= 30 incorrectly classifies C11.
    - If there are more attributes to use: Continue the process
Other Measures of Randomness

- A number of other measures can be used to produce a decision tree from a training set

- Gain Ratio = (Information Gain)/SplitInfo
  - where SplitInfo = − Σ |tᵢ|/|t| * log₂(|tᵢ|/|t|)
    - |t| is the number of entries in the table being decomposed
    - |tᵢ| is the number of entries in the iᵗʰ table produced

- Gini Index = 1 - Σ pᵢ²

Clustering

- Given:
  - a set of items
  - a set of characteristic attributes for the items
  - a similarity measure based on those attributes

- Clustering involves placing those items into clusters, such that items in the same cluster are close according to the similarity measure
  - Different from classification: there the categories are known in advance
  - For example, cancer patients might have the attribute location, and might be placed in clusters with similar locations.
Example: Clustering Students by Age

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>Id</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_1</td>
<td>17</td>
<td>17</td>
<td>3.9</td>
</tr>
<tr>
<td>S_2</td>
<td>17</td>
<td>17</td>
<td>3.5</td>
</tr>
<tr>
<td>S_3</td>
<td>18</td>
<td>18</td>
<td>3.1</td>
</tr>
<tr>
<td>S_4</td>
<td>20</td>
<td>20</td>
<td>3.0</td>
</tr>
<tr>
<td>S_5</td>
<td>23</td>
<td>23</td>
<td>3.5</td>
</tr>
<tr>
<td>S_6</td>
<td>26</td>
<td>26</td>
<td>3.6</td>
</tr>
</tbody>
</table>

K-Means Algorithm

- Center of a cluster is the mean of the items in the cluster

- To cluster a set of items into $k$ categories
  1. Pick $k$ items at random to be the (initial) centers of the clusters (so each selected item is in its own cluster)
  2. Place each item from the training set in the cluster to whose center it is the closest
  3. Recalculate the centers of each cluster
  4. Repeat the procedure starting at Step 2 until there is no change in the membership of any cluster
The Student Example (con’t)

- Suppose we want 2 clusters based on Age
  - Randomly pick S1 (age 17) and S4 (age 20) as the centers of the initial centers
  - The initial clusters are 17 17 18 20 23 26
  - The centers of these clusters are 17.333 and 23
  - Redistribute items among the clusters based on the new centers: 17 17 18 20 23 26
  - If we repeat the procedure, the clusters remain the same

<table>
<thead>
<tr>
<th>Id</th>
<th>Age</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_1</td>
<td>17</td>
<td>3.9</td>
</tr>
<tr>
<td>S_2</td>
<td>17</td>
<td>3.5</td>
</tr>
<tr>
<td>S_3</td>
<td>18</td>
<td>3.1</td>
</tr>
<tr>
<td>S_4</td>
<td>20</td>
<td>3.0</td>
</tr>
<tr>
<td>S_5</td>
<td>23</td>
<td>3.5</td>
</tr>
<tr>
<td>S_6</td>
<td>26</td>
<td>3.6</td>
</tr>
</tbody>
</table>

The Hierarchical or Agglomerative Algorithm

- Number of clusters is not fixed in advance

- Initially select each item in the training set as the center of its own cluster

- Select two clusters to merge into a single cluster
  - One approach is to pick the clusters whose centers are the closest according to some measure (e.g., Euclidian distance)

- Continue until some termination condition is reached (e.g., the number of clusters falls below some limit)
Student Example (con’t)

17  17  18  20  23  26
17  17  18  20  23  26
17  17  18  20  23  26
17  17  18  20  23  26
17  17  18  20  23  26
17  17  18  20  23  26

space complexity and time complexity?

space: $O(n^2)$ – for all pair matrix

time: $O(n^3)$ – $n$ iterations with $n^2$ updates

--- the K-means Solution

Dendrogram

- One way to manually analyze the results of the hierarchical algorithm is with the use of a tree called a **dendrogram**

- The nodes are clusters in the intermediate stages of the hierarchical algorithm

- The tree is constructed in reverse order of the execution of the hierarchical algorithm, starting with the final (single) cluster
Dendrogram for the Student Example

Analysis of Dendrogram

- Any set of nodes whose children *partition* all the leaves is a possible clustering
- For example,
  
  17  17  18  20  23  26

  is an allowable set of clusters.  
  
  *Note:* these clusters were not seen at any of the intermediate steps in the hierarchical or K-means algorithms!