Decision Tree Example

Will a patient have high-risk based on the initial 24-hour observation?
A high-risk patient usually cannot survive for more than 30 days.
**Decision Tree Example**

**Business marketing:** predict whether a person will buy a computer?

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>18</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>35</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>45</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>46</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>50</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>32</td>
<td>low</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>33</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>21</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>50</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>22</td>
<td>medium</td>
<td>yes</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>34</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>33</td>
<td>high</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>55</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>
Basic Concept

Tree is constructed by repeated splitting feature space into two subsets.

Definitions: node, terminal node (leaf node), parent node, child node.

The union of the regions occupied by two child nodes is the region occupied by their parent node.

Every leaf node is assigned with a class. A query is associated with class of the leaf node it lands it.

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rating</th>
<th>buys_computer</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>31...40</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
</tbody>
</table>
The Three Elements to Grow a Tree

• The construction of a tree involves the following three elements:
  1. The selection of the splits
  2. The decisions when to declare a node terminal or to continue splitting it
  3. The assignment of each terminal node to a class
The Three Elements

• The construction of a tree involves the following three elements:
  1. The selection of the splits
  2. The decisions when to declare a node terminal or to continue splitting it
  3. The assignment of each terminal node to a class
Standard Set of Questions

• The input vector $X = (X_1, X_2, ..., X_p)$ contains features of both categorical and ordered (numerical) types.
• Each split depends on the value of only a unique variable.
  • In the example below, the split lines are parallel to coordinates
The questions in the splitting node are in the form of
\{Is \; X_j \leq c?\} for all real-valued c.

How many candidate questions? Infinite?
The questions in the splitting node are in the form of
\{Is \ X_j \leq c?\} for all real-valued c.

Since the training data set is finite, there are only finitely many distinct splits that can be generated.
Standard Set of Questions

• Let attribute A be a continuous-valued attribute
• Must determine the *best split point c* for A
  – Sort the value A in increasing order
  – Typically, the midpoint between each pair of adjacent values is considered as a possible *split point c*
    • \((a_i + a_{i+1})/2\) is the midpoint between the values of \(a_i\) and \(a_{i+1}\)
• Split:
  – \(A \leq c\)
  – \(A > c\)
Standard Set of Questions

How about categorical features?

If $X_j$ is categorical, taking values, say in $\{1, 2, \ldots, M\}$, the questions are in the form

$$\{\text{Is } X_j \in A\}, \text{ where } A \text{ is a subset of } \{1, 2, \ldots, M\}.$$

How many candidate questions for one variable?

Exponential number of subsets: $2^M$
Heuristic efficient algorithms: see this link
Goodness of Split

• The goodness of split is measured by an impurity function defined for each node.
• Intuitively, we want each leaf node to be “pure”, that is, one class dominates.
The Impurity Function

Definition: An impurity function $\phi$ defined on the set of all $K$-tuples of numbers $(p_1, \ldots, p_K)$ satisfying $p_j \geq 0$, $j = 1, \ldots, K$, $\sum_j p_j = 1$ with the properties:

1. $\phi$ is a maximum only at the point $\left(\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K}\right)$.
2. $\phi$ achieves its minimum only at the points $(1, 0, \ldots, 0)$, $(0, 1, 0, \ldots, 0)$, $\ldots$, $(0, 0, \ldots, 0, 1)$.
3. $\phi$ is a symmetric function of $p_1$, $\ldots$, $p_K$, i.e., if you permute $p_j$, $\phi$ remains constant.

Figure and slides are from http://sites.stat.psu.edu/~jiali/course/stat557/
Definition: Given an impurity function \( \phi \), define the impurity measure \( i(t) \) of a node \( t \) as

\[
i(t) = \phi(p(1 \mid t), p(2 \mid t), ..., p(K \mid t)),
\]

where \( p(j \mid t) \) is the estimated probability of class \( j \) within node \( t \).

Goodness of a split: \( i(t) - i(t_L) - i(t_R) \)?

Left region
\( x: 8; o: 2 \)
\( i(t_L) = \Phi(0.8, 0.2) \)

Right region
\( x: 2; o: 12 \)
\( i(t_R) = \Phi(0.14, 0.86) \)

All region
\( x: 10; o: 14 \)
\( i(t) = \Phi(0.42, 0.58) \)
Definition: Given an impurity function $\phi$, define the impurity measure $i(t)$ of a node $t$ as

$$i(t) = \phi(p(1 \mid t), p(2 \mid t), \ldots, p(K \mid t)),$$

where $p(j \mid t)$ is the estimated probability of class $j$ within node $t$.

Goodness of a split $s$ for node $t$, denoted by $\Phi(s, t)$, is defined by

$$\Phi(s, t) = \Delta i(s, t) = i(t) - p_R i(t_R) - p_L i(t_L),$$

where $p_R$ and $p_L$ are the proportions of the samples in node $t$ that go to the right node $t_R$ and the left node $t_L$ respectively.

Left region
- $x$: 8; $o$: 2
- $i(t_L) = \Phi(0.8, 0.2)$

Right region
- $x$: 2; $o$: 12
- $i(t_R) = \Phi(0.14, 0.86)$

All region
- $x$: 10; $o$: 14
- $i(t) = \Phi(0.42, 0.58)$
Define \( I(t) = i(t)p(t) \), that is, the impurity function of node \( t \) weighted by the estimated proportion of data that go to node \( t \).

The impurity of tree \( T \), \( I(T) \) is defined by

\[
I(T) = \sum_{t \in \tilde{T}} I(t) = \sum_{t \in \tilde{T}} i(t)p(t). 
\]

Note for any node \( t \) the following equations hold:

\[
\begin{align*}
    p(t_L) + p(t_R) & = p(t) \\
p_L &= p(t_L)/p(t), \quad p_R = p(t_R)/p(t) \\
p_L + p_R & = 1
\end{align*}
\]

**Left region**
- \( x: 8; o: 2 \)
- \( i(t_L) = \Phi(0.8, 0.2) \)
- \( I(t_L) = i(t_L) \times 10 \)

**Right region**
- \( x: 2; o: 12 \)
- \( i(t_R) = \Phi(0.14, 0.86) \)
- \( I(t_R) = i(t_R) \times 14 \)

**All region**
- \( x: 10; o: 14 \)
- \( i(t) = \Phi(0.42, 0.58) \times 24 \)
Define

\[ \Delta l(s, t) = \int(0.42, 0.58) - \int(0.8, 0.2) - \int(0.14, 0.86) \]

\[ = \int(0.42, 0.58) * 24 - \int(0.8, 0.2) * 10 - \int(0.14, 0.86) * 14 \]

All region
\[ x: 10; o: 14 \]
\[ i(t) = \Phi(0.42, 0.58) \]
\[ l(t) = i(t) * 24 \]

Left region
\[ x: 8; o: 2 \]
\[ i(t_L) = \Phi(0.8, 0.2) \]
\[ l(t_L) = i(t_L) * 10 \]

Right region
\[ x: 2; o: 12 \]
\[ i(t_R) = \Phi(0.14, 0.86) \]
\[ l(t_R) = i(t_R) * 14 \]

\[ \Delta l(s, t) = \Phi(0.42, 0.58) * 24 - \Phi(0.8, 0.2) * 10 - \Phi(0.14, 0.86) * 14 \]
\[ = 24 * (\Phi(0.42, 0.58) - \Phi(0.8, 0.2) * (10/24) - \Phi(0.14, 0.86) * (14/24)) \]
The Impurity Function: Entropy

Entropy: \( \sum_{j=1}^{K} p_j \log \frac{1}{p_j} \). If \( p_j = 0 \), use the limit 
\( \lim_{p_j \to 0} p_j \log p_j = 0 \).

Algorithms using Entropy:
ID3 (Interactive Dichotomiser 3)
C4.5 (successor of ID3)
The Impurity Function: Gini index

Gini index: \[ \sum_{j=1}^{K} p_j (1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2 \]

Algorithms using Gini index:

CART: Classification And Regression Tree

1. Entropy: \( \sum_{j=1}^{K} p_j \log \frac{1}{p_j} \). If \( p_j = 0 \), use the limit \( \lim_{p_j \to 0} p_j \log p_j = 0 \).
2. Misclassification rate: \( 1 - \max_j p_j \).
3. Gini index: \( \sum_{j=1}^{K} p_j(1 - p_j) = 1 - \sum_{j=1}^{K} p_j^2 \).

Consider 400 cases in each class
One split option:
(300, 100) and (100, 300)

Another split option:
(200, 400) and (200, 0)

Error rate treats them the same
Entropy and Gini index prefers latter

Node impurity measures for binary classification. Entropy has been rescaled.
The Three Elements

• The construction of a tree involves the following three elements:
  1. The selection of the splits
  2. The decisions when to declare a node terminal or to continue splitting it
  3. The assignment of each terminal node to a class
Stopping Criteria

1. The samples in a node have the same labels
2. The goodness of split is lower than a threshold
3. The number of samples in every leaf is below a threshold
4. The number of leaf nodes reaches a limit
5. Tree has exceeded the maximum desired depth
Stopping Criteria

- A simple criteria: stop splitting a node $t$ when

$$\max_{s \in S} \Delta I(s, t) < \beta,$$

where $\beta$ is a chosen threshold.

- The above stopping criteria is unsatisfactory.
  - A node with a small decrease of impurity after one step of splitting may have a large decrease after multiple levels of splits.
In this example, the first split is always not good. Using stopping criteria based on threshold, it will will not split.
The Three Elements

• The construction of a tree involves the following three elements:
  1. The selection of the splits
  2. The decisions when to declare a node terminal or to continue splitting it
  3. The assignment of each terminal node to a class
Class Assignment Rule

- A class assignment rule assigns a class \( j = \{1, \ldots, K\} \) to every terminal node \( t \in \widetilde{T} \). The class assigned to node \( t \in \widetilde{T} \) is denoted by \( \kappa(t) \).
- For 0-1 loss, the class assignment rule is:

\[
\kappa(t) = \arg \max_j p(j \mid t).
\]
The Three Elements to Grow a Tree

• The construction of a tree involves the following three elements:
  1. The selection of the splits
     • Gini index (CART)
     • Entropy (ID3, C4.5)
     • Misclassification error
  2. The decisions when to declare a node terminal or to continue splitting it
  3. The assignment of each terminal node to a class
Example: Iris Flower Data

Iris setosa  Iris versicolor  Iris virginica

4 features
• Sepal length
• Sepal width
• Petal length
• Petal width

Iris data. We only show the first two features, sepal length and sepal width.
Murphy, Figure 16.5(a), dtreeDemolris

Full decision tree
Example: Diabetes data

- Diabetes data
- 768 samples
- 8 quantitative variables
- 2 classes

R script: https://onlinecourses.science.psu.edu/stat857/node/133
Decision tree in R: http://cran.r-project.org/web/packages/rpart/rpart.pdf
The Three Elements to Grow a Tree

• The construction of a tree involves the following three elements:

  1. The selection of the splits
     • Gini index (CART)
     • Entropy (ID3, C4.5)
     • Misclassification error

  2. The decisions when to declare a node terminal or to continue splitting it
     • grow to a full tree and then prune it

  3. The assignment of each terminal node to a class
Bias and Variance

<table>
<thead>
<tr>
<th>(high) Bias</th>
<th>(high) Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>High training error</td>
<td>High testing error</td>
</tr>
<tr>
<td>Under-fit</td>
<td>Over-fit</td>
</tr>
<tr>
<td>Simple model</td>
<td>Complex model</td>
</tr>
</tbody>
</table>

Examples:
- An over-pruned tree
- One-degree linear

Examples:
- A “full” tree
- A high-degree polynomial linear regression
Bias and Variance

<table>
<thead>
<tr>
<th>(high) Bias</th>
<th>(high) Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>High training error</td>
<td>High testing error</td>
</tr>
<tr>
<td>Under-fit</td>
<td>Over-fit</td>
</tr>
<tr>
<td>Simple model</td>
<td>Complex model</td>
</tr>
</tbody>
</table>

Examples:
- An over-pruned tree
- One-degree linear

Examples:
- A “full” tree
- A high-degree polynomial linear regression
Pruning a Tree

• To prevent overfitting, we can stop growing the tree, but the stopping criteria tends to be myopic

• The standard approach is therefore to grow a “full” tree, and then to perform pruning
  – We can use any of the previous stopping criteria with very low threshold to grow a “full” tree (a very large tree)
How to Evaluate a Tree (or generally a classification model)?

- Randomly select a portion of data as training (e.g., 80%)
- Remaining data as testing (e.g., 20%)

- Then, compare different trees (or models)
How to Evaluate a Tree (or generally a classification model)?

Random partition could be biased. Cross validation is preferred.

Example: 5-fold cross validation

Final result to report:
- Average accuracy over 5 rounds
- Standard deviation of accuracy over 5 rounds
Parameter Tuning by Random Partition of Train/Test

1. Randomly choose a portion as train and the remaining as test
   1. Tree starts with one (terminal) node (m=1)
   2. Train a tree with m terminal nodes
   3. Test it and get the error rate
   4. Grow the tree with one more split (m ← m + 1)
   5. If the tree is not full yet, go to 2
2. Pick m* terminal nodes which give us the lowest error
Parameter Tuning by Cross-Validation

Random partition could be biased. Cross validation is preferred.

1. Use k-fold cross-validation on training data
2. In each validation, \((k-1)/k\) of training data is used as train, \(1/k\) of training data is used as test
   1. Tree starts with one (terminal) node \((m=1)\)
   2. Train a tree with \(m\) terminal nodes
   3. Test it and get the error rate
   4. Grow the tree with one more split \((m \leftarrow m + 1)\)
   5. If the tree is not full yet, go to 2
3. Pick \(m^*\) terminal nodes which give us the lowest error

5-fold cross validation
Evaluating a Tree with the “Best” Parameter

Randomly pick 80% as train, 20% as test (if using 5-fold cross validation, need to do this process 5 times)

Use m* terminal nodes to train a model on all training data

Pick m* terminal nodes

Note: there are 2 layers of train/test splitting!!!
Murphy, Figure 16.6, dtreeDemoIris
Murphy, Figure 16.6, dtreeDemoIris
Example: Diabetes data

Pick the minimum cross-validation error
Nsplit = 5
# of terminal nodes = 6

R script: https://onlinecourses.science.psu.edu/stat857/node/133
Pros of Trees

- Easy to interpret
- Handle mixed discrete and continuous inputs
- Insensitive to monotone transformations of the inputs
  - Because the split points are based on ranking the data points
- Perform automatic variable selection
- Relatively robust to outliers
- Scale well to large data sets
- Can be modified to handle missing features
Missing Values

• Certain variables are missing in some training samples
  – Often occurs in gene-expression microarray data
  – Suppose each variable has 5% chance being missing independently. Then for a training sample with 50 variables, the probability of missing some variables is as high as 92.3%

• A sample to be classified may have missing variables

• Find surrogate splits
  – Suppose the best split for node $t$ is $s$ which involves a question on $X_m$. Find another split $s'$ on a variable $X_j, j \neq m$, which is most similar to $s$ in a certain sense. Similarly, the second best surrogate split, the third, and so on, can be found
  – Advantage over a generative model: not modeling the entire joint distribution of inputs
  – Disadvantage: entirely ad-hoc

• Code “missing” as a new value for categorical variables
Cons of Trees

• Do not predict very accurately compared to other kinds of model
  – Due to the greedy nature of the tree construction algorithm

• Trees are unstable
  – Small changes to the input data can have large effects on the structure of the tree
    • Due to the hierarchical nature of the tree-growing process
  – Solution: bagging and boosting
Summary

• Decision tree
  – CART: Classification And Regression Tree

• Grow a tree
  – Selection of splits
  – Stopping criteria
  – Class assignments in leaf nodes

• Prune a tree
  – Grow to a “full” tree
  – Use cross-validation to prune