Decision Tree Learning

Don't be afraid of decision tree learning!

- Simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed.
- Able to handle both numerical and categorical data.
- Uses a white box model.

Section 1. Classification task

Medical diagnosis example

- **Real world objects:** heart attack patients admitted to the hospital.
- **Feature vector:** age and medical symptoms indicating the patient's conditions (got by measurements), like blood pressure.
Output values: high risk patients (G) and not high risk patients (F).

Goal: Identification of G- and F- patients on the basis of measurements.

1. Like a doctor, based on your experience, try to formulate criteria to identify the patients:
   - 1st question: Is the minimum systolic blood pressure over the initial 24 hour period greater than 91?
     - If "no", then the patient is not high risk patient.
     - If "yes", 2nd question: Is age greater than 62.5?
     - If "no", then the patient is not high risk patient.
     - If "yes", 3rd question: ...

Let's visualize it.

Medical diagnosis problem: classification rules

Credits: Leo Breiman et al.

2. Like a machine learning expert, design a classifier based on the training data.

Section 2. Tree structured classifiers

A tree is a graph $G = (V, E)$ in which any two vertices are connected by exactly one simple path.

A tree is called a rooted tree if one vertex has been designated the root, in which case the edges have a natural orientation, towards or away from the root.
A tree in graph theory: any two vertices are connected by exactly one simple path.

A rooted tree

A decision tree $G_{DT} = (V_{DT}, E_{DT})$ is a rooted tree composed of internal decision nodes and terminal leaf nodes.
Section 3. A tree-based classifier construction

corresponds to building decision tree $G^{X}_{DT} = (V^{X}_{DT}, E^{X}_{DT})$ based on a data set $X$. A decision node $t \in V_{DT}$ is a subset of $X$ and the root node $t_1 = X$. Each leaf node is designated by an output value (i.e. class label).

$G^{X}_{DT} = (V^{X}_{DT}, E^{X}_{DT})$ corresponds to repeated splits of subsets of $X$ into descendant subsets, beginning with $X$ itself.

It is necessary to specify:

1. The set $Q$ of questions

Assume that the feature vectors have the form $x = \langle x_1, x_2, ..., x_M \rangle$. The standardized set of questions $Q$ is defined as follows:

- Each split depends on the value of only a SINGLE feature.
- For each quantitative feature $x_m$, $Q$ includes all the questions of the form Is $x_m \leq c$? for all $c$ ranging over $(-\infty, +\infty)$.
- For each qualitative feature $x_m$ having values, say, in $\{b_1, b_2, ..., b_L\}$, then $Q$ includes all questions of the form Is $x_m \in Q_S$ as $Q_S = 2^{\{b_1, b_2, ..., b_L\}}$.

An alternative way of looking at the decision tree splits:

Each decision node $t$ implements a test function $f_t(x)$ ($\langle x, y \rangle \in t$) with discrete outcomes labeling the branches.

Each $f_t(x)$ defines a discriminant in the $M$-dimensional input space dividing it into small regions which are further subdivided as we take a path from the root down.
2. **A rule for selecting the best split**

**Fundamental idea**: select each split of a subset so that the data in each of the descendant subsets are "purer" than the data in the parent node. Be \( c \) a number of output values (i.e. classes):

1. **Define** the node proportions \( p(j|t), j = 1, \ldots, c, \) to be the proportion of the instances \( x: \langle x, j \rangle \in D \wedge \langle x, j \rangle \in t, \) so that \( \sum_{j=1}^{c} p(j|t) = 1. \)

2. **Define** a measure \( i(t) \) of the impurity of \( t \) as a nonnegative function \( \Phi \) of the \( p(1|t), p(2|t), \ldots, p(c|t) \) such that
   - \( \Phi(\frac{1}{c}, \frac{1}{c}, \ldots, \frac{1}{c}) = \max \) i.e. the node impurity is largest when all examples are equally mixed together in it.
   - \( \Phi(1, 0, \ldots, 0) = 0, \Phi(0, 1, \ldots, 0) = 0, \ldots, \Phi(0, 0, \ldots, 1) = 0 \) i.e. the node impurity is smallest when the node contains only one class.

3. **Define** the goodness of split to be the decrease in impurity
   \( \Delta i(s, t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R)p_L \) is a proportion of examples in \( t \) going to \( t_L, \) similarly with \( p_R. \)
4. Define a candidate set $S$ of splits at each node: the set $Q$ of questions generates a set $S$ of splits $s$ of every node $t$. Find split $s^*$ with the largest decrease in impurity:
\[ \Delta i(s^*, t) = \max_{s \in S} \Delta i(s, t). \]

5. Discuss four splitting criteria: Misclassification Error, Information Gain, Gain Ratio and Gini Index.

In general, a classification output takes values $1, 2, ..., c$ and
\[ i(t) = \hat{\Phi}(p(1|t), p(2|t), ..., p(c|t)). \]

**Misclassification error**

The node impurity is defined by
\[ i(t) = \frac{1}{|t|} \sum \delta_{y_i, \hat{k}} = 1 - p_{\hat{k}}^t, \]
where
\[ \hat{k} = \arg\max_k p_k^t, p_k^t = \frac{1}{|t|} \sum \delta_{y_i, k}, \]
i.e. $\hat{k}$ is the majority class in node $t$.

**Information gain**

The node impurity is defined by
\[ i(t) = - \sum_j p(j|t) \log(p(j|t)). \]

**Note:** There is convincing justification for this specific form of $i(t)$. It was selected simply because it was a familiar function having the properties required by step 2 above. Recall the notion of entropy, then $i(t) = H(t)$.

\[ \text{Gain}(s, t) = \Delta i(s, t) = i(t) - \sum_{s_j} p_j \cdot i(t_j) = H(t) - \sum_{s_j} \frac{|t_j|}{|t|} H(t_j). \]

Other notation used in ML community: $\text{Gain}(s, t) = \text{Gain}(D, A)$
The information gain favors features with many values over those with few values (many values, many branches, impurity can be much less).

**Example**

Let’s consider the feature *Date* (October 19, 2010). It has so many values that it is bound to separate the training examples into very small subsets. As it has a high information gain relative to the training data, it is probably selected for the root node. But prediction behind the training examples is poor. Let’s incorporate an alternative measure **Gain ratio** that takes into account the number of values of a feature.

**Gain ratio**

\[
\text{GainRatio}(s, t) = \frac{\text{Gain}(s, t)}{\text{SplitInformation}(s, t)},
\]

where

\[
\text{SplitInformation}(s, t) = -\sum_{s_j} \frac{|t_j|}{|t|} \log_2 \frac{|t_j|}{|t|},
\]

\[
t_j = \{ (x, y) \in t; A(x) = j \}, |\text{Values}(A) = c|
\]

**SplitInformation** is the entropy of \( t \) with respect to the split \( s \), i.e. to the values of feature \( A \).

**Gini index**

The node impurity is defined by

\[
i(t) = \sum_{j \neq i} p(j|t)p(i|t)
\]

(also written as \( i(t) = 1 - \sum_j p^2(j|t) \)). In the two-class problem \( i(t) = 2 * p(1|t) * p(2|t) \).

Interpretation: use the rule that assigns an instance selected at random from the node to class \( i \) with probability \( p(i|t) \). The estimated probability that the item is actually in class \( j \) is \( p(j|t) \). The estimated probability of misclassification is the Gini index. In other words, Gini can be interpreted as expected error rate.

**Misclassification vs. Entropy vs. Gini**

For two classes (i.e. \( c = 2 \)), if \( p \) is the proportion in the class "1", the measures are:

- Misclassification error: \( 1 - \max(p, 1 - p) \)
- Entropy: \( -p \log p - (1 - p) \log(1 - p) \)
- Gini: \( 2p * (1 - p) \)

Figure 8
A two-class problem: Misclassification vs. Entropy vs. Gini

Note: Figure generated by R script posted at http://ufal.mff.cuni.cz/~hladka/lab/entropy-gini-miss.R.

- All three measures are similar, but Entropy and Gini are differentiable, and hence more subject to numerical optimization.
- Entropy and Gini are more sensitive to changes in the node probabilities than the misclassification error rate.

**Example:**

A two-class problem. 800 training examples: (+400, -400). Two splits:

- $t_L^1: (+300, -100)$, $t_R^1: (+100, -300)$; after split-2
- $t_L^2: (+200, -400)$, $t_R^2: (+200, 0)$. Both splits produce a misclassification error of 0.25 (Check it!). The second split produces a pure node and is probably preferable. Both Gini and Entropy are lower for the second split. Thus either Gini or Entropy should be used when growing the tree.

3. **A criterion for choosing the right sized tree**

- When a node $t$ was reached such that no significant decrease in impurity was possible (i.e. $\max_{s} \Delta i(s, t) < \beta$), then $t$ was not split and became a terminal node. The class of this terminal node is determined as follows: if $p(j_0|t) = \max_j p(j|t)$, then $j_0$ becomes a class of this terminal node. But ...
  - If $\beta$ is set too low, then there is too much splitting and the tree is too large.
  - Increasing $\beta$: there may be nodes $t$ such that $\max_{s} \Delta i(s, t) < \beta$ is small. But the descendants nodes $t_L$ and $t_R$ of $t$ may have splits with large decreases in impurity. By declaring $t$ terminal, one looses the good splits on $t_L$ or $t_R$.

- Prune instead of stopping - see below

**Section 4. Decision tree learning**

Decision tree learning is a data-driven method for modelling discrete-valued predicted function in which the learned hypothesis is represented by a decision tree.
Learning mechanism - ID3 algorithm

Notation

- $A_c$ is a set of output values to be predicted by the tree,
- $Attr$ is a list of features used,
- $Values(A)$ is a set of all possible values for feature $A$,
- $D^A_v$ is the subset of $D$ for which feature $A$ has value $v$, i.e., $D^A_v = \{(x, y) \in D | A(x) = v\}$.

Figure 9

ID3 notation in colors

ID3($D, A_c, Attr$)

- Create a $Root$ node for the tree.
- If $\forall x, (x, y) \in D : y = 1$, return single node tree $Root$ with label = ‘+’
- If $\forall x, (x, y) \in D : y = -1$, return single node tree $Root$ with label = ‘-’
- If $Attr = \emptyset$, return a single node tree $Root$ with label = most common value of $A_c$ in $D$, i.e. $n = \max \sum_{v \in Values(A_c)} |(x, y) \in D \delta(v, A_c(x))|$
- Otherwise
  - $A := \text{Splitting Criterion}(Attr)$
  - $Root := A$
  - $\forall v \in Values(A)$
    - Add a new tree branch below $Root$, corresponding to the test $A(x) = v$
    - If $D_v = \emptyset$
      - then below this new branch add a leaf node with label = $n = \max \sum_{v \in Values(A_c), |(x, y) \in D \delta(v, A_c(x))|}$
      - else below this new branch add the subtree $ID3(D_v, A_c, Attr - \{A\})$
  - End
ID3 is a recursive partitioning algorithm (divide & conquer), performs top-down tree construction.

ID3 performs a simple-to-complex, hill-climbing search through $H$ guided by a particular splitting criterion.

ID3 maintains only a single current hypothesis (recall CE algorithm). So ID3 for example is not able to determine any other decision trees consistent with training data.

ID3 does not employ backtracking.

ID3 uses all training examples at each step (recall Find-S and CE) to make statistically-based decision. That is why the search is not so sensitive to errors in the training examples.

Section 5. Avoiding over-fitting the data

Definition

Given a hypothesis space $H$, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that $h$ has smaller error than $h'$ over the training examples, but $h'$ has a smaller error than $h$ over the entire distribution of instances.

In context of DT: too many branches, some may reflect anomalies due to noise or outliers.

Figure 10

Overtraining: decision tree case

Credits: Tom Mitchell

Example


$$D = \{(2, 2), (+), (3, 3), (+), (-2, -2), (+), (-3, -3), (+), (-2, 2), (-), (-3, 3), (-), (2, -2), (-), (3, -3), (-)\}$$
Pruning aims to simplify those decision trees that overfitted the data. Both top-down and bottom-up methods proposed. Some methods use the training data to evaluate the accuracy of a decision tree, others exploit an additional pruning set.

**Pruning techniques**

A branch of tree $T$ contains a node $t$. Then all its descendants will be indicated as $T_t$.

- Reduced error $p$.

  The complete tree $T_{max}$. REP uses the pruning set to evaluate the efficacy of a subtree of $T_{max}$. For each internal node of $T_{max}$, it compares the number of classification errors made on the pruning set when the subtree $T_t$ is kept, with the number of classification errors made when $t$ is turned into a leaf and associated with the best class. This method is of linear computational complexity, since each node is visited only once. I.e.

  - Classify examples in pruning set – some might be errors
  - For each node:
    - Sum the errors over entire subtree
    - Calculate error on same example if converted to a leaf with majority class label
    - Prune node with highest reduction in error
    - Repeat until error no longer reduced

**Example**

Figure 14
Reduced-error pruning

- Pessimistic error p.
- Minimum error p.
- Critical value p.
- Cost complexity p.
- Error-based p.

Implemented in C4.5 (an extension of the ID3 algorithm)

Rule post-pruning
- Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
- Prune each rule by removing any preconditions that result in improving its estimated accuracy.
- Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

Section 6. Incorporating continuous-valued attributes

ID3 is originally designed with two restrictions:

1. Discrete-valued predicted feature.
2. Discrete-valued features tested in the decision tree nodes. --> Let's extend it for the continuous-valued features.

For a continuous-valued features $A$ define a boolean-valued feature $A_c$ so that if $A(x) \leq c$ then $A_c(x) = 1$ else $A_c(x) = 0$.

How to select the best value for the threshold $c$?

Criterion: Choose such $c$ that produces the greatest information gain.

Many heuristics, for example: sort the examples according to continuous feature $A$ and identify examples that differ in their classification. It can be shown that the value of $c$ that maximizes information gain must always lie at a such boundary.
<table>
<thead>
<tr>
<th>Temperature</th>
<th>20</th>
<th>22</th>
<th>24</th>
<th>26</th>
<th>28</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>EnjoySport</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

\[ c_1 = \frac{22 + 24}{2}, \quad c_2 = \frac{28 + 30}{2} \]

\[ \text{Gain}(\text{Temperature} \geq c_1, D) = ?, \quad \text{Gain}(\text{Temperature} \geq c_2, D) = ? \]

**Section 7. Handling training examples with missing feature values**

Consider the situation in which \( \text{Gain}(D, A) \) (i.e. \( \text{Gain}(s, t) \)) is to be calculated at node \( n \) in the decision tree. Suppose that \( \langle x, c(x) \rangle \) is one of the training examples in \( D \) and that the value \( A(x) \) is unknown.

**Possible solutions:**

- Assign the value that is most common among training examples at node \( n \).
- Alternatively, assign the most common value among examples at node \( n \) that have the classification \( c(x) \).
- Alternatively, in this case, the classification of the new instance is simply the most probable classification, computed by summing the weights of the example fragments classified in different ways at the leaf nodes of the tree. This method for handling missing feature values is used in C4.5.

**Section 8. The strength of DT methods are:**

- Robust to noisy data.
- Capable of learning disjunctive expressions.
- Decision trees are able to generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are able to handle both continuous and categorical features.
- Decision trees provide a clear indication of which features are most important for classification.

**Section 9. The weaknesses of DT methods are:**

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous feature (--> regression tree considered as a variant of decision trees, designed to predict real-valued functions instead of being used for classification task).
- Decision trees are prone to errors in classification problems with many class and relatively small number of training examples.
- Decision tree can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting feature must be sorted before its best split can be found. In some algorithms, combinations of features are used and a search must be made for optimal combining weights. Pruning algorithms can also be expensive since many candidate sub-trees must be formed and compared.
- Decision trees do not treat well non-rectangular regions. Most decision-tree algorithms only examine a single feature at a time. This leads to rectangular classification boxes that may not correspond well with the actual distribution of records in the decision space.

**Section 10. Final remarks**

Many, many extensions of ID3 algorithm.

Supervised learning algorithms degrade in performance when faced with many features that are not necessary for predicting the desired output. **How to select a good subset of features?**
A good choice of features may not only help improve performance, but also aid in finding smaller models for the data, resulting in better understanding and interpretation of the data.

**Decision trees** should be sufficiently complex to account for the known facts, but at the same time it should be as simple as possible.

**Pruning**

Initially, many nodes are often pruned going from one tree to the next smaller tree in the sequence, but fewer nodes tend to be pruned as the root node is approached.

It will certainly cause decrease of accuracy on the training set. However it may increase the accuracy on an independently chosen test set.

**Low/High Cost Features**

The features may have associated costs. For ex. medical diseases classification, features such as temperature, BiopsyResult, Pulse, BloodTestResults, etc. They vary in their costs, both in terms of monetary cost and cost to patient comfort. In such tasks, low-cost features are preferable; relying on high-cost features only when to produce reliable classifications.