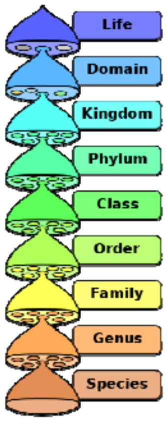
Lecture 10: Decision Tree and Random Forrest

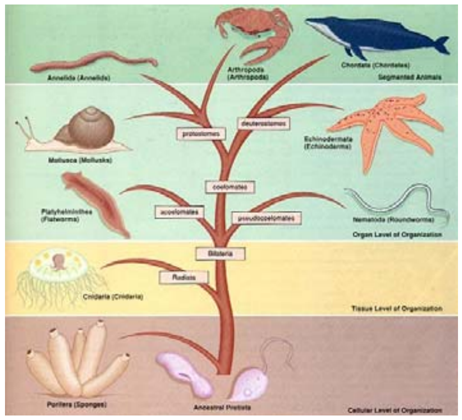
In some applications of pattern classification, such as plant/animal categorization, medical diagnosis, car insurance etc. tree-structured classification methods are found to be more intuitive and effective. Tree structured and rule-based classifiers have been used in these applications for a long time, before they were brought to the attention of the pattern recognition community by L. Breiman et al. in a 1984 book on Classification And Regression Tree. The method is called CART.

In recent years, decision trees have been widely used for designing weak classifiers, which are then combined in Boosting (like Adaboost and LogitBoost), and Random Forrest.

Examples of Tree Structured Classification

Pictures from wikipedia



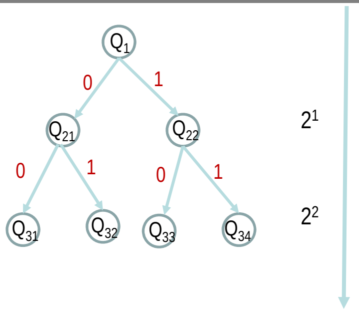


The game of 20 Questions

The intuitive idea is similar to the game of 20-questions. The objective is to design an effective and balanced binary tree, each non-terminal node is a question, a test, or a rule, with 2 possible outcomes.

After 20 questions, the tree will have about
$2^{20} = 10^6$
a million leaf nodes. Hopefully any objects/concepts that a person can conceive belong to one of the leaf Node, and thus is localized.

If a leaf note contains more than 1 object/concept, then they cannot be told apart.



General formulation for decision tree

Given a set of labeled training data

$$\Omega = \{(x_i, c_i) : \ x_i \in \Omega^d, c_i \in \{1,2,...,K\}, \ i = 1,2,...,N\}$$

Objective:

1. Construct a set of tests (like the weak classifiers { h() } in Adaboost)

$$\Delta = \{s_i : \ s_i(x) \in \{1,2,...,K_i\}, \ i = 1,2,...,M\}$$

2. Combine a subset of selected tests to build a tree T (like a strong classifier H in Adaboost).

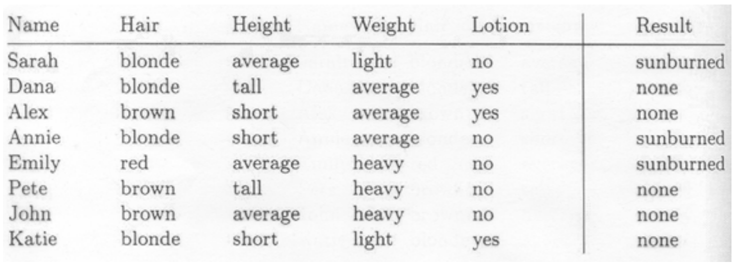
Each step a test is chosen to maximally reduce an impurity measure of the tree.
The impurity of the tree measures the empirical error on the training set.

A Simple Example: training data

In this example, we have N=8 input examples with K=2 classes: {sunburned, none}.
That is to predict/classify whether a person gets sunburned based on 4 features

x = (hair color, height, weight, using lotion)

$$\Omega = \{(x_i, c_i):\ x_i \in \Omega^d, c_i \in \{1, 2, ..., K_i\},\ i = 1, 2, ..., N\}$$

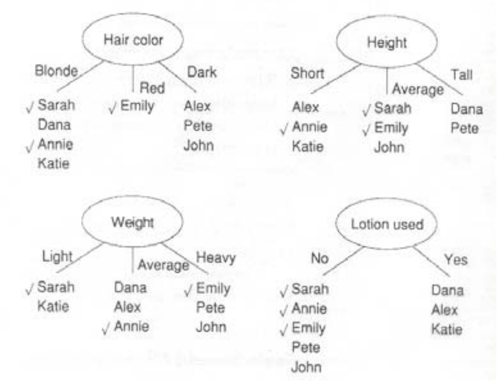


Name	Hair	Height	Weight	Lotion	Result
Sarah	blonde	average	light	no	sunburned
Dana	blonde	tall	average	yes	none
Alex	brown	short	average	yes	none
Annie	blonde	short	average	no	sunburned
Emily	red	average	heavy	no	sunburned
Pete	brown	tall	heavy	no	none
John	brown	average	heavy	no	none
Katie	blonde	short	light	yes	none

A Simple Example: tests

$$\Delta = \{s_i:\ s_i(x) \in \{1, 2, ..., K\},\ i = 1, 2, ..., M\}$$

Below are some simple tests based on a single feature. Clearly the lotion feature is more informative than weight. How do we measure the information?



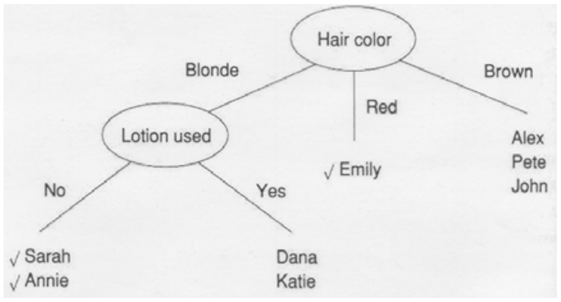
[One possible measure is mutual information between a test result s and the label y:

$$MI(s, y) = \int p(s, y) \log \frac{p(s, y)}{p(s)p(y)}\, dxdy$$

]

A Simple Example: a short decision tree

This decision tree T only uses 2 tests to separate all the N=8 training samples.
This tree is said to be pure with respect to the training data.



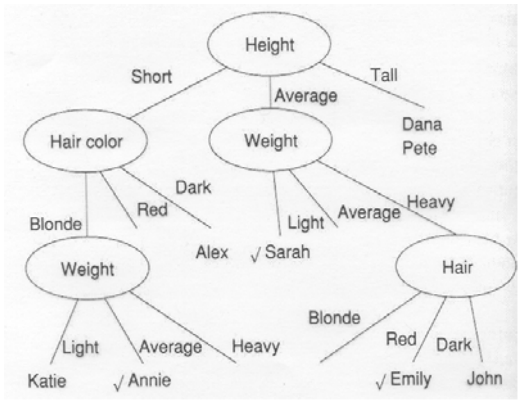
In the decision tree, the tests are sequentially applied, thus

1, They only apply to a sub-population, and are "local".

2, They are combined in a non-linear way, in contrast to the linear sum in boosting.

A Simple Example: a bad tree

This decision tree T uses 5 tests to separate all the N=8 training samples.
It is less effective than the previous tree.



In both cases, it is not hard to make the training error =0, i.e. each leaf node has instance(s) of the same class --- The node is said to be pure, i.e. not mixed.

Questions:
1, Isn't this a greedy way to construct the decision tree?
2, Isn't it overfitting?
3, Isn't the structure of the tree unstable?

Yes, Yes, Yes.

Tree Structured Classification

Given a set of labeled samples, a tree classier recursively divides the set of samples in a leaf node of the tree into two subsets, until a certain stopping rule is observed.

Major components in tree structured classification.

1. The criteria for splitting.

2. The design of tests.

3. The rule for stopping and tree pruning.

4. The rule for labeling the leaves of the tree.

5. Data validation, variable combination, and missing data strategy.

Criteria for Splitting

The Criteria for splitting a leaf node is to make the population of the samples in the children nodes purer than the current leaf.

Given a node t in the tree T, $p(j \mid t)$; $j = 1, 2, \ldots, k$ is the proportion (probability) of training samples labeled j in t.

In general, the impurity of a leaf node is an entropy of $p(j \mid t)$.
There are two typical choices of entropy

1. The Shannon entropy

$$i(t) = -\sum_{j=1}^{K} p(j|t) \log p(j|t)$$

2. The Gini-index

$$i(t) = 1 - \sum_{j=1}^{K} p(j|t)^2$$

The Criteria for Splitting

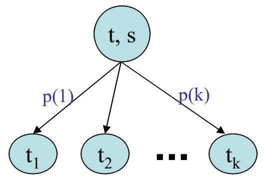
The impurity of the entire tree T is the weighted sum of impurities of its leaves.

$$I(T) = \sum_{t \in L(T)} i(t)p(t)$$

where L(T) is the set of all leaves in the tree T, and p(t) is the proportion of the training samples in t (i.e. the more examples are in a leaf, the heavier weight it has). When this impurity is zero, then the empirical (training) error is also zero.

To build a classification tree, we start with one root node as the leaf including all the training examples. Each time we choose to split one leaf node *t* by a certain test *s* so that the impurity of the tree decreases the most.

$$\Delta i(s,t) = i(t) - \sum_{j=1}^{K} p(j)i(t_j)$$



Tree Structured Classification

The change (decrease) of impurity of the tree is the impurity change weighted by the population of leaf node t.

$$\Delta I(s,t) = I(T) - I(T_+) = \Delta i(s,t)p(t)$$

Therefore the choice of the next best leaf node t and testing rule s is

$$(s,t)^* = \arg \max_{s \in \Delta, t \in L(T)} \Delta I(s,t)$$

The Design of Tests

Now how to design the set S of candidate questions ? Enumerating all combinations is impractical.

Let $x = (x_1, x_2, \ldots, x_n)$ be the features for a pattern, the tests or questions are in general designed for a single feature.

If x_i is an ordered variable, the question set is

$\{$ is $x_i > c$? for any $c\}$

If x_i is categorical, the candidate questions are

$\{$ is x_i in A? for any $A\}$

The above standard questions result in rectangular partitions of the feature space. We can design more complex questions by combining the variables.

$\{$ is $a_i x_i + a_j x_j > c$? $\}$

Boolean expression for binary variables.

$\{$ is $x_1 > 0.5$ and $x_3 < 0.2$? $\}$

The Stopping Rule and Surrogate Split

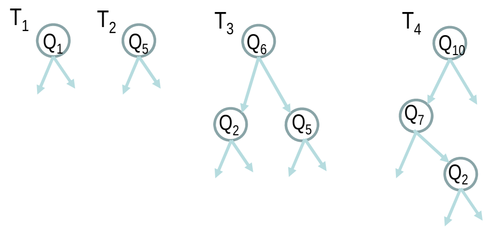
If one keeps splitting the tree, finally each leaf contains only one sample or samples of the same label. This is not necessarily a good tree for the test set. It is again a model complexity problem that we met before. One may stop splitting a node above certain impurity threshold. Then the terminal leaf is labeled by the label of majority.

The complexity of the decision tree is usually decided by cross-validation, i.e. choosing a tree structure that has good performance on the test set.

One may treat various decision trees as weaker classifiers and feed them to the Adaboost.

The missing data problem is solved by an engineering technique called *surrogate split*. Define a measure of two split rules s and s' of a leaf t. If the best split of t is the split rule s based on feature x_m, find the split s' on the variables other than x_m that is most similar to s. Call s' the best surrogate split.

Random Forrest



Instead of constructing one giant tree, people found that it is often more robust to build a large set of dwarf trees, and thus obtain a forrest, and then combine them by majority voting.

To make these trees less dependent of each other (i.e. to diversify), we can train each tree by
1, Taking a random subset of the training data, sometimes by Bootstrapping (replaced sampling).
2, Taking a subset of the designed tests so that some strong tests won't dominate.

Then we count the votes: $T_1(x) + T_2(x) + \ldots + T_n(x)$

Random Forrest

Suppose each tree is a binary classifier: $\mathrm{T}(\mathrm{x}) \in \{-1, \ +1\}$

Then we count the votes:

$$F(x) = T_1(x) + T_2(x) + \ldots + T_n(x)$$

If each tree is a multi-class classifier: $\mathrm{T}(\mathrm{x}) \in \{1, 2, \ldots, \mathrm{K}\}$

Then we count the votes for class k by

$$1(T_1(x)=k) + 1(T_2(x)=k) + \ldots + 1(T_n(x)=k)$$

1() is the indicator function.