Computing for Data Sciences

Lecture 14 – 15

Decision Tree is a machine learning algorithm for classification of data. It builds classification models in the form of a tree structure which gets constructed by classifying data into classes based on a series of questions about the attributes of the data. The dataset is consequently broken down into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node has two or more branches while leaf nodes which branch out from the corresponding decision nodes represent a classification or decision. The topmost decision node in a tree which corresponds to the best predictor is called **root node**. Decision trees can handle both categorical and numerical data while the classes are always categorical in nature.

The objective of asking questions about the attributes of the data is to have the maximum information gain about the class of the data. Right questions provide the maximum information gain and a method of quantifying this information is through the notion of Shannon's entropy.

Shannon's Entropy

Entropy is a function quantifying the uncertainty associated with a random variable.As uncertainty and or randomness increases for a data set so does the entropy and the value ranges from 0 – 1 to represent the entropy of information.

A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous) or at least trying to achieve as much homogeneity as possible. The decision tree algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided it has entropy of one. For e.g., take a coin, if we know the coin is completely biased towards heads or tails, then we know the outcome of toss with complete certainty and hence there is no information gain. But if we had a perfectly unbiased coin, then the entropy would be maximum.

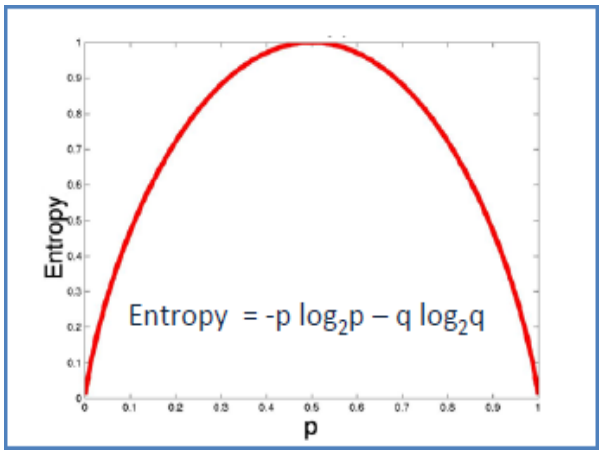


Fig 1 : Variation of entropy with probability

1. The entropy of sample in a node is calculated as:

$$H = -\sum p_i \log_2 p_i$$

where p_i is probability of i^{th}class.

In the above example, if coin is unbiased i.e., when the class of the data is unknown then $p_{heads} = 0.5 = p_{tails}$ which gives H = 1. This is the case when data is equally divided. But in case of completely homogeneous data i.e., when we actually know the class (heads or tails), the entropy H = 0.

2. The above formula was for a particular node. To find the entropy in all the child nodes in a certain level in the decision tree, we need to find the expectation of H which is given by:

$$E(H) = \sum \frac{n_i}{n} H(c_i)$$

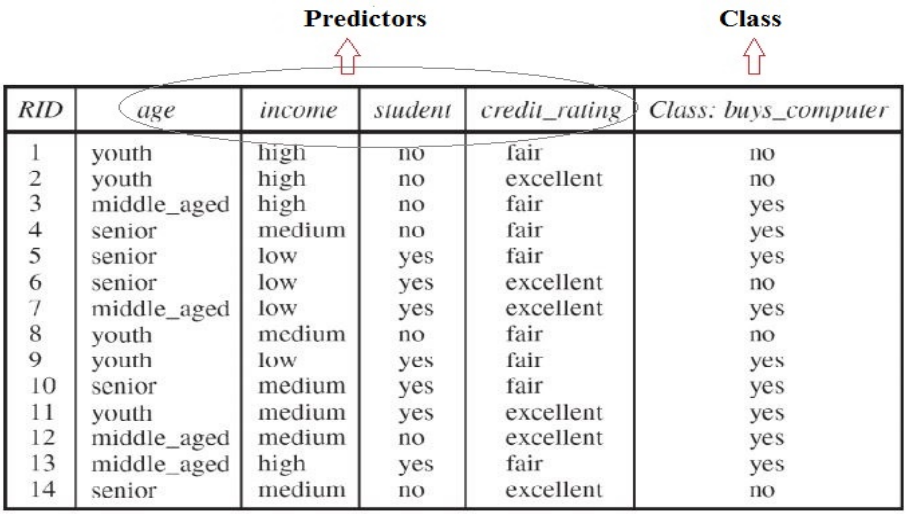
where
c_i is the i^{th}child node,
n_i is the number of elements in the i^{th} child node,
n is the number of elements in just the upper node or the decision node, $\sum n_i = n$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding the attribute or asking the question about the attribute that returns the highest information gain (i.e., the most homogeneous branches). For simplicity, the partitioning is done based on a single attribute. But we can do the partition based on a combination of attributes as well.

$$Information\ gain = E(decision\ node\ entropy) - E(child\ node\ entropy)$$

Example: Training data



RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Fig 2: Training data

Root node: P_{yes}=9/14, P_{no}=5/14

Step 1: H = - (9/14)$\log_2$(9/14) – (5/14)$\log_2$(5/14) = 0.940

Step 2: Compute the expected information requirement for each attribute: start with the attribute age

H(youth) = -(2/5)$\log_2$(2/5) – (3/5)$\log_2$(3/5) = 0.9709506

H(middle_aged) = 0

H(senior) = – (3/5)$\log_2$(3/5) - (2/5)$\log_2$(2/5) = 0.9709506

E(child node entropy) = (5/14) H(youth) + (4/14) H(middle_aged) + (5/14) H(senior)

= 0.6935361
Information gain (age) = 0.940 - 0.6935361
= 0.246

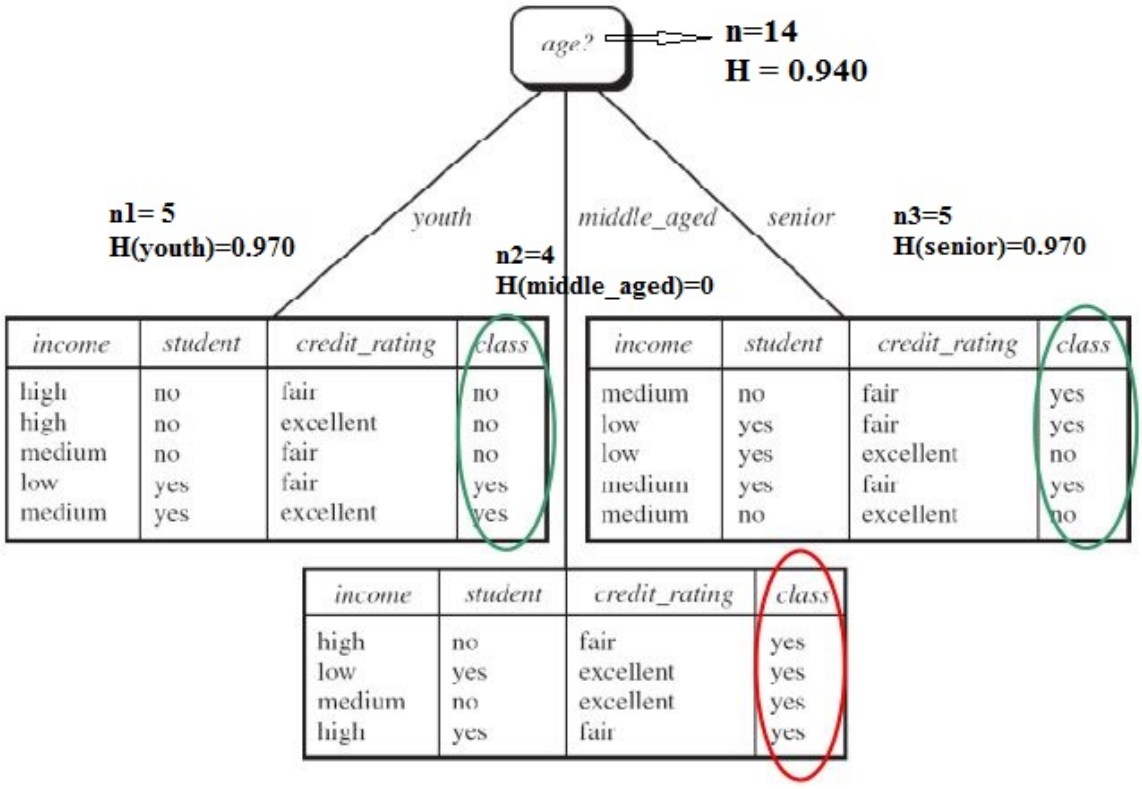


Fig 3 : Information gain due to “age” attribute

Similarly, if we calculate information gain for other attributes, we get the following:

Information gain (income) = 0.029
Information gain (student) = 0.151
Information gain (credit_rating) = 0.048

We can similarly find out which attributes to use for further partitioning until the leaf nodes are reached .

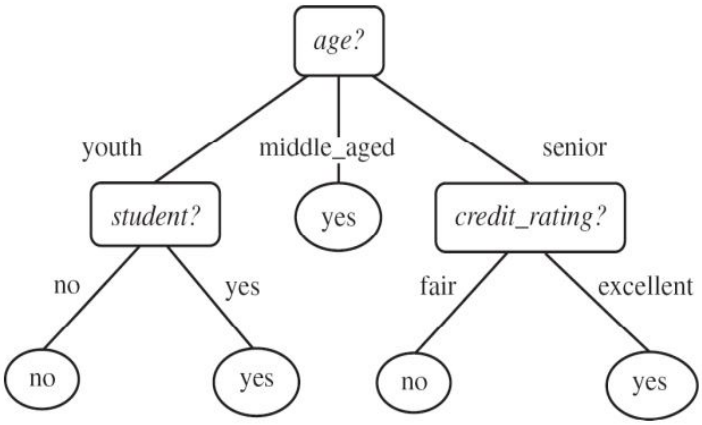


Fig 4 : A final decision tree for the above problem would look somewhat like this

Random Forests

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of over fitting to their training set.

Tree Bragging

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set X = x1, ..., xn with responses Y= y1, ..., yn, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For b = 1, ..., B:

1. Sample, with replacement, n training examples from X, Y; call these Xb, Yb.
2. Train a decision or regression tree fb on Xb, Yb.

The Decision Forrest Model

A random decision forest is an ensemble of randomly trained decision trees. The forest model is characterized by a number of components. For instance, we need to choose a family of split functions (also referred to as “weak learners” for consistency with the literature). Similarly, we must select the type of leaf predictor. The randomness model also has great influence on the workings of the forest.

The Weak Learner Model

Each split node j is associated with a binary split function,

$$h(v, \vartheta j) \in \{0, 1\},$$

with e.g. 0 indicating “false” and 1 indicating “true”. The data arriving at the split node is sent to its left or right child node according to the result of the test. The weak learner model is characterized by its parameters $\theta = (\phi, \psi, \tau)$ where ψ defines the geometric primitive used to separate the data (e.g. an axis-aligned hyper plane, an oblique hyper plane, a general surface etc.). The parameter vector τ captures thresholds for the inequalities used in the binary test. The filter function ϕ selects some

features of choice out of the entire vector v. All these parameters will be optimized at each split node.

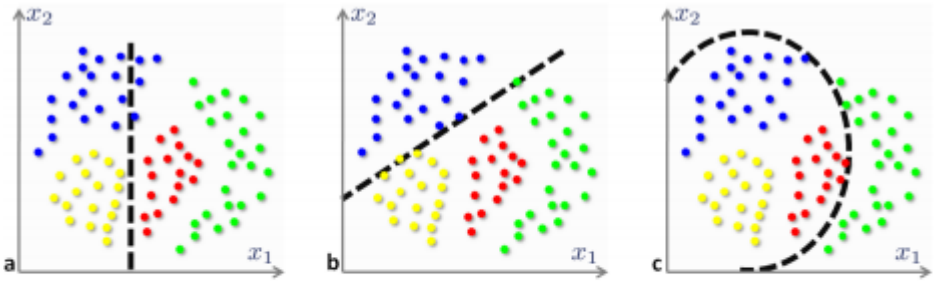


Fig. 2.6: **Example weak learners. (a)** Axis-aligned hyperplane. **(b)** General oriented hyperplane. **(c)** Quadratic (conic in 2D). For ease of visualization here we have $\mathbf{v} = (x_1\ x_2) \in \mathbb{R}^2$ and $\phi(\mathbf{v}) = (x_1\ x_2\ 1)$ in homogeneous coordinates. In general data points $\mathbf{v}$ may have a much higher dimensionality and ϕ still a dimensionality of ≤ 2.

Source : http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf

The Training Objective Function

During training, the optimal parameters θ_j^* of the j th split node need to be computed. This is done here by maximizing an information gain objective function:

$$\boldsymbol{\theta_j^* = arg\ max_{\theta_j}\ I_j}$$

with

$$\boldsymbol{I_j = I(S_j\ ,\ S_j^L,\ S_j^R\ ,\ \theta_j\ ).}$$

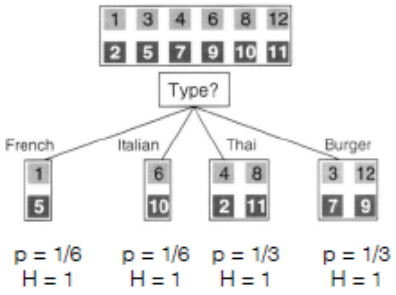
The symbols $\boldsymbol{S_j\ ,\ S_j^L,\ S_j^R\ ,\ \theta_j}$ denote the sets of training points before and after the split. The Equation is of an abstract form here. Its precise definition depends on the task at hand (e.g. supervised or not, continuous or discrete output).

Example for split:

Training data:

Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	$$$	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	$	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	$	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	$	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	$$$	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	$	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	$$	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	$	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$$$	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	$	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	$	*No*	*No*	*Burger*	*30–60*	*Yes*

From the given training data,
People waited: 1, 3, 4, 6, 8, and 12 (six)
People did not wait: 2, 5,7,9,10,11 (six)



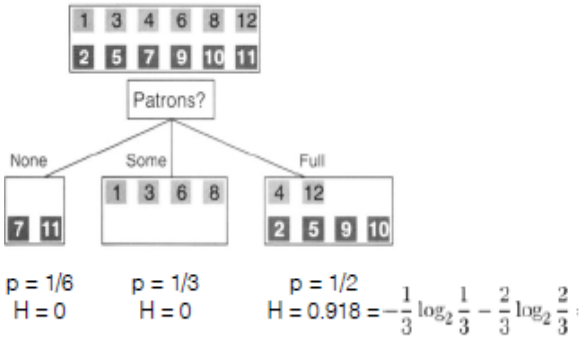
Information gain by using "Type" parameter as splitting node is Zero.
= H- E(H) = 1 -1 = 0

Parent Entropy

$$H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

E(Child Entropy)

$$E(H) = \frac{1}{6}\times 1 + \frac{1}{6}\times 1 + \frac{1}{3}\times 1 + \frac{1}{3}\times 1 = 1$$



Parent Entropy

$$H = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$$

E(Child Entropy)

$$E(H) = \frac{1}{6}\times 0 + \frac{1}{3}\times 0 + \frac{1}{2}\times 0.918 = 0.459$$

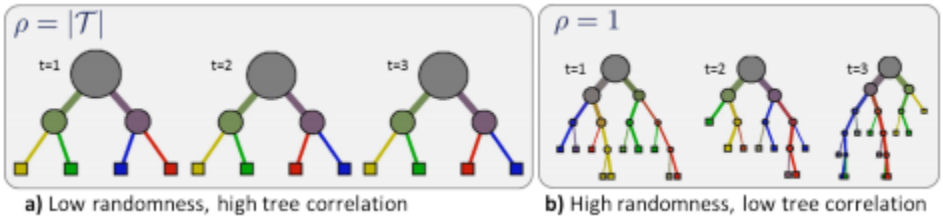
Information gain by using "Patron" parameter as splitting node is 0.541

= H- E(H) = 1 -0.459 = 0.541

So for splitting at this node, "Patron" parameter is more optimal than "Type" parameter.

Node optimization

The maximization operation can be achieved simply as an exhaustive search operation. Often, finding the optimal values of the τ thresholds may be obtained efficiently by means of integral histograms.



a) Low randomness, high tree correlation **b)** High randomness, low tree correlation

Fig. 2.7: **Controlling the amount of randomness and tree correlation. (a)** Large values of ρ correspond to little randomness and thus large tree correlation. In this case the forest behaves very much as if it was made of a single tree. **(b)** Small values of ρ correspond to large randomness in the training process. Thus the forest component trees are all very different from one another.

Source : http://research.microsoft.com/pubs/155552/decisionForests_MSR_TR_2011_114.pdf

If parameters for tree construction are similar for all trees, ρ will larger. Large randomness will result in weak correlation between trees of the forest.

Let us consider a data which has 10000 observations, and each observation has information about 1000 features. At each node of splitting, parameters for splitting at each node are selected randomly and optimal parameters are computed from these selected parameters.

The Randomness Model

A key aspect of decision forests is the fact that its component trees are all randomly different from one another. This leads to de-correlation between the individual tree predictions and, in turn, to improved generalization. Forest randomness also helps achieve high robustness with respect to noisy data. Randomness is injected into the trees during the training phase. Two of the most popular ways of doing so are:

• random training data set sampling (e.g. bagging), and
• randomized node optimization.

The Ensemble Model

In a forest with T trees we have t ∈ {1, · · · , T}. All trees are trained independently (and possibly in parallel). During testing, each test point v is simultaneously pushed through all trees (starting at the root) until it reaches the corresponding leaves. Tree testing can also often be done in parallel, thus achieving high computational efficiency on modern 17 parallel CPU or GPU hardware.

Small note on decision tree algorithms:

For more details on methods on decision tree classifiers : http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf

1) C4.5

C4.5 is an evolution of ID3. The ID3 algorithm is considered as a very simple decision tree algorithm. ID3 uses information gain as splitting criteria. The growing stops when all instances belong to a single value of target feature or when best information gain is not greater than zero. ID3 does not apply any pruning procedures nor does it handle numeric attributes or missing values. C4.5 uses gain ratio as splitting criteria. The splitting ceases when the number of instances to be split is below a certain threshold. Error–based pruning is performed after the growing phase. C4.5 can handle numeric attributes. It can induce from a training set that incorporates missing values by using corrected gain ratio criteria as presented above.

2) CART

CART stands for Classification and Regression Trees. It is characterized by the fact that it constructs binary trees, namely each internal node has exactly two outgoing edges. An important feature of CART is its ability to generate regression trees. Regression trees are trees where their leaves predict a real number and not a class. In case of regression, CART looks for splits that minimize the prediction squared error (the least–squared deviation). The prediction in each leaf is based on the weighted mean for node.

Metrics for measure of quality of split:

https://en.wikipedia.org/wiki/Decision_tree_learning

For choosing a variable at each step that best splits the set of items, Different algorithms use different metrics for measuring "best". These generally measure the homogeneity of the target variable within the subsets. These metrics are applied to each candidate subset, and the resulting values are combined (e.g., averaged) to provide a measure of the quality of the split.

1) Information gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute.

$$H = -\sum p_i \log_2 p_i$$

2) GINI Index

Used by the CART (classification and regression tree) algorithm, Gini entropy is a measure of how often a randomly chosen element from the set would be incorrectly labelled if it were randomly labelled according to the distribution of labels in the subset. Gini entropy can be computed by summing the probability p_i of each item being chosen times the probability of $1 - p_i$ a mistake in categorizing that item. It reaches its minimum (zero) when all cases in the node fall into a single target category.

To compute Gini entropy for a set of items, suppose, and p_i let be the fraction of items labelled with value ith in the set.

$$I_G(p) = \sum_{i=1}^{m} p_i(1-p_i) = \sum_{i=1}^{m} p_i - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} p_i^2 = \sum_{\substack{i \neq k \\ for\ all\ m}}^{m} p_i p_k$$

3) Variance reduction

Introduced in CART, variance reduction is often employed in cases where the target variable is continuous (regression tree), meaning that use of many other metrics would first require discretization before being applied. The variance reduction of a node is defined as the total reduction of the variance of the target variable due to the split at the node.

Summary of key model Parameters

In summary, the parameters that most influence the behaviour of a decision forest are:

- The forest size T;
- The maximum allowed tree depth D;
- The amount of randomness (controlled by ρ) and its type;
- The choice of weak learner model;
- The training objective function;
- The choice of features in practical applications.

Those choices directly affect the forest predictive accuracy, the accuracy of its confidence, its generalization and its computational efficiency.