Classification: Decision Tree

Introduction to Artificial Intelligence with Mathematics
Lecture Notes

Ganguk Hwang

Department of Mathematical Sciences
KAIST

Decision Trees Structure

- Decision Trees (DTs) are a supervised learning method used for classification and regression.
- The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- A decision tree is a simple representation for classifying data.
- Decision tree learning is one of the most successful techniques for supervised classification learning.

Decision Tree Classification

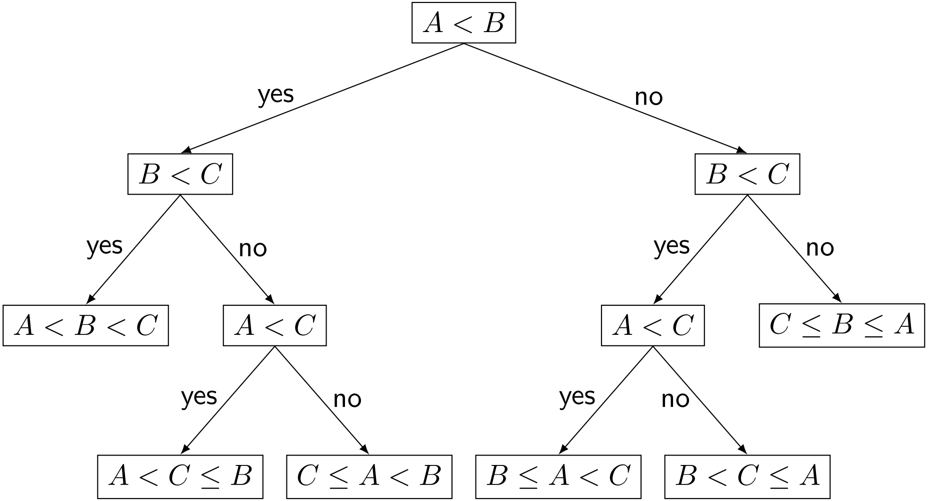


Figure: A decision tree for sorting three values.

A **decision tree** consists of the following:

- **A root node** has no incoming edges and two or more outgoing edges
- **Internal nodes** have exactly one incoming edge and two or more outgoing edges. Each internal node is a new question.
- **Leaf/terminal nodes** have exactly one incoming edge and no outgoing edges. Each leaf node corresponds to a classification conclusion.

The input of a machine learning algorithm consists of a set of **instances** (e.g. rows, examples or observations).

Each instance is described by

- a fixed number of **attributes** (i.e. columns), which are assumed to be either nominal or numeric, and
- a label which is called **class** (in case of a classification task). The set of all instances is called **dataset**.

Instance	Attribute			Class
	$A < B$	$B < C$	$A < C$	
1	yes	yes	yes	$A < B < C$
2	yes	yes	no	$A < B < C$
3	yes	no	yes	$A < C \leq B$
4	yes	no	no	$C \leq A < B$
5	no	yes	yes	$B \leq A < C$
6	no	yes	no	$B < C \leq A$
7	no	no	yes	$C \leq B \leq A$
8	no	no	no	$C \leq B \leq A$

Table: Dataset table for the sorting example

Decision Tree Learning

In what follows we see how to generate a decision tree from a given dataset in general.

- Splitting: choose the best attribute for classification
- Pruning: avoid overfitting

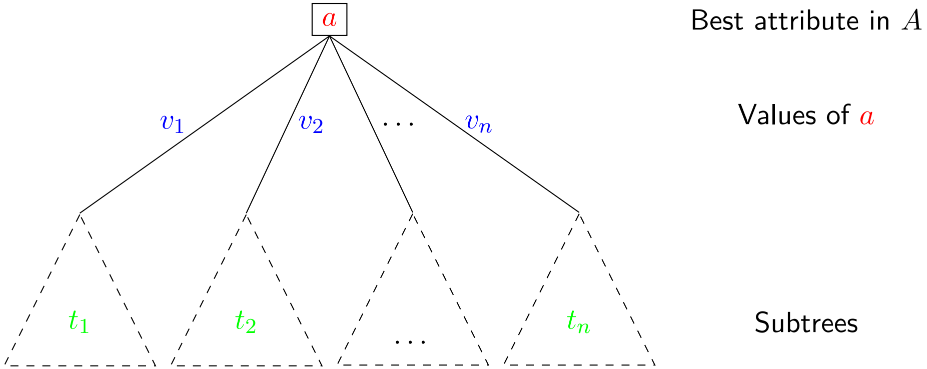


Figure: The basic structure of a decision tree

Splitting Criterion

- To choose the best attribute, we need to decide a criterion to split the data set.
- All criteria are based on *impurity* measure. It defines how well classes are separated.

- We assume that there are a total number of C classes denoted by $\Omega = \{\omega_1, \omega_2, \dots, \omega_C\}$.
- Let there be N training data values represented by

$$(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$$

where x_i is a vector of n attributes and $y_i \in \Omega$ is the class label corresponding to the input x_i.

- Of these N training data values, N_{ω_k} data values belong to the class ω_k, while $\sum_k N_{\omega_k} = N$.
- When we use attribute a_j, it splits the training data set into P partitions, or P child nodes, each of which has $N^{(p)}$ data values.
- The number of class ω_k data values in a particular partition p is denoted by $N_{\omega_k}^{(p)}$. Then $\sum_k N_{\omega_k}^{(p)} = N^{(p)}$.

Recall that the entropy $H(X)$ of X is defined by

$$H(X) = -E[\log p(X)] = E\left[\log\left(\frac{1}{p(X)}\right)\right].$$

Information Gain:

The idea of the information gain is based on the information theory.

Definition 1

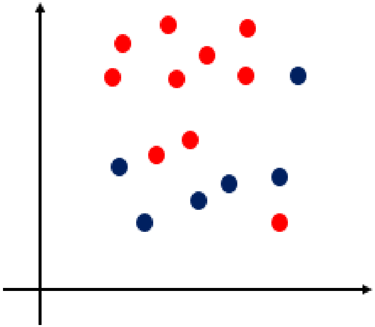
*The **information gain** for attribute a_j is defined by*

$$G(a_j) = \left(\sum_{k=1}^{C} -\frac{N_{\omega_k}}{N} \log_2 \frac{N_{\omega_k}}{N} \right) - \left(\sum_{p=1}^{P} \frac{N^{(p)}}{N} \cdot \sum_{k=1}^{C} -\frac{N_{\omega_k}^{(p)}}{N^{(p)}} \log_2 \frac{N_{\omega_k}^{(p)}}{N^{(p)}} \right)$$

- The first term is the entropy of the training data set.
- The second term the weighted entropy of the child nodes.
- The difference thus reflects the decrease in entropy or the information gained from the use of attribute a_j.
- cf. $I(X;Y) = H(X) - H(X|Y)$

Consider the following example with two types - red and blue. The entropy H is computed as follows:

$$H = -\frac{10}{16}\log_2\left(\frac{10}{16}\right) - \frac{6}{16}\log_2\left(\frac{6}{16}\right) \approx 0.95.$$

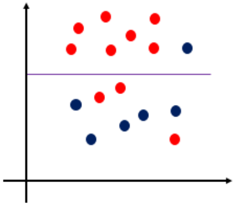


When the data set is divided into two subgroups as given below, the weighted sum of two entropies are computed as follows:

$$\begin{aligned} H_{new} &= 0.5 \times \left\{ -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \right\} \\ &\qquad + 0.5 \times \left\{ -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right) \right\} \\ &\approx 0.75. \end{aligned}$$

So the information gain in this case is

$$G = H - H_{new} \approx 0.95 - 0.75 = 0.2.$$

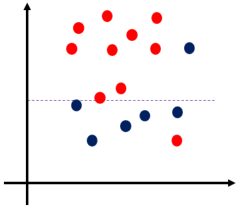


When we use a new line as given below to divide the data set into two subgroups, the weighted sum of two entropies are computed as follows:

$$\begin{aligned} H_{new} = &\frac{10}{16} \times \left\{ -\frac{9}{10} \log_2 \left(\frac{9}{10} \right) - \frac{1}{10} \log_2 \left(\frac{1}{10} \right) \right\} \\ &+ \frac{6}{16} \times \left\{ -\frac{1}{6} \log_2 \left(\frac{1}{6} \right) - \frac{5}{6} \log_2 \left(\frac{5}{6} \right) \right\} \\ \approx &\, 0.54. \end{aligned}$$

So the information gain in this case is

$$G = H - H_{new} \approx 0.95 - 0.54 = 0.41.$$



Gini Index (or Gini Impurity)

- The Gini Index or Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.
- The Gini impurity can be computed by summing the probability p_i of an item with label i being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing the item.

For a random variable X with pmf $p_1, \cdots, p_n$, the Gini Index (GI) is defined by

$$GI = \sum_{i=1}^{n} p_i(1 - p_i) = 1 - \sum_{i=1}^{n} p_i^2.$$

The GI reaches its minimum (zero) when all data values fall into a single target category. For instance, when $p_i = 1$ for some i and $p_j = 0$ for all $j \neq i$, then

$$GI = 1 - (0^2 + \cdots + 0^2 + 1^2 + 0^2 + \cdots + 0^2) = 0$$

which can be easily understood as that the impurity is 0.

To select the best attribute, we consider the following.

For the training data set, we have

$$p_k = \frac{N_{\omega_k}}{N} \text{ and } GI_{org} = 1 - \sum_{k=1}^{C} \left(\frac{N_{\omega_k}}{N} \right)^2 .$$

When we use attribute a_j which splits the training data set into P subsets, for subset p, we have

$$p_k^{(p)} = \frac{N_{\omega_k}^{(p)}}{N^{(p)}} \text{ and } GI_p(a_j) = 1 - \sum_{k=1}^{C} \left(\frac{N_{\omega_k}^{(p)}}{N^{(p)}} \right)^2$$

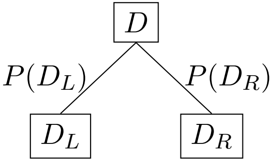
The gain $G_{GI}(a_j)$ from attribute a_j can be computed as

$$
\begin{aligned}
G_{GI}(a_j) &= GI_{org} - \sum_{p=1}^{P} \frac{N^{(p)}}{N} GI_p(a_j) \\
&= \sum_{p=1}^{P} \frac{N^{(p)}}{N} \sum_{k=1}^{C} \left(\frac{N_{\omega_k}^{(p)}}{N^{(p)}} \right)^2 - \sum_{k=1}^{C} \left(\frac{N_{\omega_k}}{N} \right)^2 .
\end{aligned}
$$

Twoing Criterion

$$T = P(D_L) \cdot P(D_R) \cdot \left(\sum_{\omega_k} |P(\omega_k|D_L) - P(\omega_k|D_R)| \right)^2 \qquad (0.1)$$

where $P(D_L)$ and $P(D_R)$ are the probability of going left or right, respectively, and $P(\omega_k|D_L)$ and $P(\omega_k|D_R)$ are the proportions of data values in D_L and D_R which belong to class ω_k.



We take the attribute a_j that maximizes the value T.

Chi-Squared Statistic Criterion

The formula for computing the χ^2 value is

$$\chi^2 = \sum_{k=1}^{C} \sum_{p=1}^{P} \frac{\left(N_{\omega_k}^{(p)} - \tilde{N}_{\omega_k}^{(p)}\right)^2}{\tilde{N}_{\omega_k}^{(p)}}$$

where $\tilde{N}_{\omega_k}^{(p)} = \frac{N_{\omega_k}}{N} \cdot N^{(p)}$ is the a priori frequency of the samples N in ω_k.

A larger value of χ^2 indicates that the split is more homogeneous, i.e., has a greater frequency of data values from a certain class. The attribute is chosen by the largest value of χ^2.

Remark: The Pearson Chi-square Test Statistic

Let $X_j, 1 \leq j \leq n$ be i.i.d. (independent and identically distributed) random variables taking their values on $\{1, 2, ..., k\}$. We are interested in testing the following null hypothesis H_0

$$H_0 : P\{X = i\} = p_i,\ 1 \leq i \leq k$$

To do so, let N_i be the number of the X_j's that equal i. Since X_j are i.i.d,

$$N_i \sim B(n, p_i).$$

Hence, when H_0 is true, $E[N_i] = np_i$. Now consider

$$TS = \sum_{i=1}^{k} \frac{(N_i - np_i)^2}{np_i} \quad \left(= \sum_{i=1}^{k} \frac{(N_i - E[N_i])^2}{E[N_i]} \right).$$

It can be shown that, for sufficiently large n the distribution of TS is very close to a χ^2 distribution with $k-1$ degrees of freedom.

For instance, when $k = 2$,

$$\begin{aligned}
TS &= \sum_{i=1}^{2} \frac{(N_i - np_i)^2}{np_i} \\
&= \frac{(N_1 - np_1)^2}{np_1} + \frac{(n - N_1 - n(1 - p_1))^2}{n(1 - p_1)} \\
&= \frac{(N_1 - np_1)^2}{np_1} + \frac{(N_1 - np_1)^2}{n(1 - p_1)} \\
&= \frac{(N_1 - np_1)^2}{np_1(1 - p_1)} \\
&= \left(\frac{N_1 - np_1}{\sqrt{np_1(1 - p_1)}} \right)^2 \approx \chi_1^2.
\end{aligned}$$

Stopping Criterion

The growing phase continues until a stopping criterion is triggered. The following conditions are common stopping rules.

- All data values in the training set belong to a single label.
- The maximum tree depth has been reached.
- The number of cases in the terminal node is less than the minimum number of cases for parent nodes.
- The best splitting criterion is not greater than a certain threshold.

The last stopping criterion is used in the implementation for pruning after the growing phase.

Tree Pruning

- Using a tight stopping criterion tends to create small and underfitted decision trees.
- On the other hand, using loose stopping criterion tends to generate large decision trees that are overfitted to the training set.
- To avoid both extremes, the idea of pruning was developed: A loose stopping criterion is used, and after the growing phase the overfitted tree is cut back into a smaller tree by removing sub-branches that are not contributing to the generalization accuracy.

Reduced Error Pruning

- The data set is divided into two subsets: training set and validation set.
- Consider each node for pruning that removes the subtree at that node, makes it a leaf node, and assigns the most common class at that node.
- A node is removed if the resulting tree performs no worse than the original tree on the validation set, which removes coincidences and errors.
- Nodes are removed iteratively by choosing the node whose removal most increases the decision tree accuracy.
- Pruning continues until further pruning is harmful.

Cost-Complexity Pruning

- Cost-Complexity Pruning is proposed by Leo Breiman in 1984.
- The pruning method is also known as weakest link pruning or error-complexity pruning.
- Contrary to the reduced error pruning this pruning approach is not straight forward.

The idea is to consider the size and the estimated error rate of the tree. The total cost $C_\alpha(T)$ of tree T is defined as

$$C_\alpha(T) = R(T) + \alpha|\tilde{T}|, \qquad \alpha \geq 0$$

where $R(T)$ is the weighted summed error of the leaves of tree T and α is penalty for the complexity of the tree $\tilde{T}$ (so called complexity parameter), while $\tilde{T}$ stands for the set of all leaves in the tree T.

Algorithms for Decision Trees

- Automatic Interaction Detection (AID)
- Chi-squared Automatic Interaction Detector (CHAID)
- Classification And Regression Tree (CART)
- Iterative Dichotomiser 3 (ID3)
- C4.5

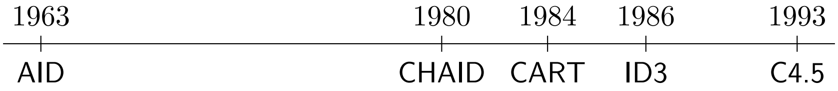


Figure: Timeline of four basic algorithm families: CHAID, CART, ID3 and C4.5

AID and CHAID

- In 1964, the statisticians John A. Sonquist and James N. Morgan introduced a first version of tree learning algorithm called Automatic Interaction Detection (AID).
- Gordon V. Kass proposed a modification to AID in his paper from 1979 called *Chi-squared Automatic Interaction Detector* (CHAID).
- The basic improvement was splitting criterion: instead of the variance the χ^2-test is used. Therefore, the partitions need not be a bisection.
- CHAID handles missing values by treating them as a separate valid category and does not perform any pruning.

CART (Classification and Regression Tree)

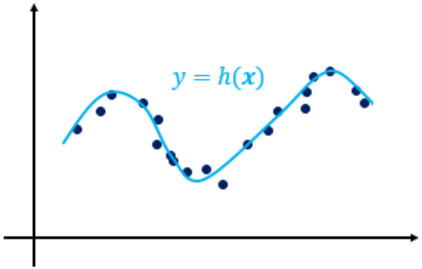
- Leo Breiman published his idea of CART in 1984.
- Two splitting criteria can be chosen: the twoing or Gini criterion. The obtained tree is pruned by cost-complexity pruning.
- L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, Classification And Regression Trees, Chapman & Hall/CRC, 1984.

CART: Regression Tree

For a given data set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)\}$ we want to construct a fucntion $f(\mathbf{x})$ such that

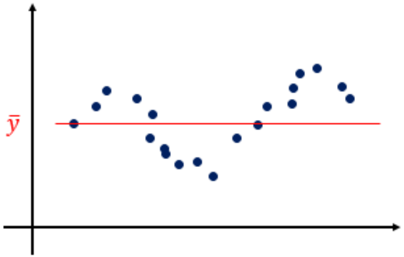
$$\sum_{i=1}^{N} |f(\mathbf{x}_i) - y_i|^2$$

is sufficiently small.



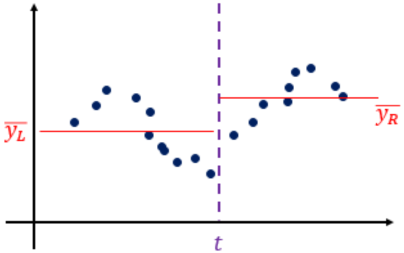
We start with the sample average $\bar{y}$.

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$$



For a threshold value t that divides the data set into two subgroups $\mathcal{D}_L$ and $\mathcal{D}_R$, we compute the sample averages $\bar{y}_L$ and $\bar{y}_R$ as follows:

$$\bar{y}_L = \frac{1}{|\mathcal{D}_L|} \sum_{i:\mathbf{x}_i \in \mathcal{D}_L} y_i, \quad \bar{y}_R = \frac{1}{|\mathcal{D}_R|} \sum_{i:\mathbf{x}_i \in \mathcal{D}_R} y_i.$$



Note that, for

$$\begin{aligned}
\text{MSE}(\mathcal{D}) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2, \\
\text{MSE}(\mathcal{D}_L) &= \frac{1}{|\mathcal{D}_L|}\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i - \bar{y}_L)^2, \\
\text{MSE}(\mathcal{D}_R) &= \frac{1}{|\mathcal{D}_R|}\sum_{i:\mathbf{x}_i\in\mathcal{D}_R}(y_i - \bar{y}_R)^2,
\end{aligned}$$

we have

$$\text{MSE}(\mathcal{D}) \geq \frac{|\mathcal{D}_L|}{N}\text{MSE}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{N}\text{MSE}(\mathcal{D}_R).$$

To see the inequality, first observe that for any value y

$$\begin{aligned}
\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-y)^2 &= \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-\bar{y}_L+\bar{y}_L-y)^2 \\
&= \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-\bar{y}_L)^2 + 2\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-\bar{y}_L)(\bar{y}_L-y) \\
&\qquad + \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(\bar{y}_L-y)^2 \\
&= \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-\bar{y}_L)^2 + \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(\bar{y}_L-y)^2 \\
&\geq \sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i-\bar{y}_L)^2.
\end{aligned}$$

Then the inequality is proved as follows:

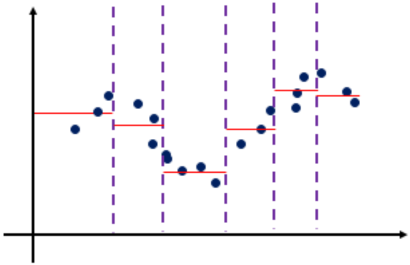
$$\begin{aligned}
\mathsf{MSE}(\mathcal{D}) &= \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \\
&= \frac{1}{N}\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i - \bar{y})^2 + \frac{1}{N}\sum_{i:\mathbf{x}_i\in\mathcal{D}_R}(y_i - \bar{y})^2 \\
&\geq \frac{1}{N}\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i - \bar{y}_L)^2 + \frac{1}{N}\sum_{i:\mathbf{x}_i\in\mathcal{D}_R}(y_i - \bar{y}_R)^2 \\
&\geq \frac{|\mathcal{D}_L|}{N}\frac{1}{|\mathcal{D}_L|}\sum_{i:\mathbf{x}_i\in\mathcal{D}_L}(y_i - \bar{y}_L)^2 + \frac{|\mathcal{D}_R|}{N}\frac{1}{|\mathcal{D}_R|}\sum_{i:\mathbf{x}_i\in\mathcal{D}_R}(y_i - \bar{y}_R)^2 \\
&= \frac{|\mathcal{D}_L|}{N}\mathsf{MSE}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{N}\mathsf{MSE}(\mathcal{D}_R).
\end{aligned}$$

We take a threshold t that maximizes the difference

$$\mathsf{MSE}(\mathcal{D}) - \left(\frac{|\mathcal{D}_L|}{N}\mathsf{MSE}(\mathcal{D}_L) + \frac{|\mathcal{D}_R|}{N}\mathsf{MSE}(\mathcal{D}_R)\right).$$

We do the division procedure until stopping criterion is satisfied.

The resulting function $f(\mathbf{x})$ is a piecewise linear function as shown below.



ID3

The ID3 algorithm was developed by Ross Quinlan in 1986. It uses the information gain as splitting criterion and it does not apply any pruning procedure nor does it handle numeric attributes or missing values.

- The ID3 algorithm begins with the original set $\mathcal{D}$ as the root node.
- On each iteration of the algorithm, it iterates through every unused attribute of the set $\mathcal{D}$ and calculates the entropy $G(a)$.
- It then selects the attribute which has the smallest entropy (or largest information gain) value. The set $\mathcal{D}$ is then split or partitioned by the selected attribute to produce subsets of the data.
- The algorithm continues to recur on each subset, considering only attributes never selected before.

C4.5

- C4.5 is the updated version of ID3 by the same author.
- One of the problems in ID3 is that the information gain criterion favors large number of partition P.
- The new splitting criterion in C4.5 is called gain ratio $\frac{G(a_j)}{g}$ where

$$g = -\sum_{p=1}^{P} \frac{N^{(p)}}{N} \log_2 \frac{N^{(p)}}{N}.$$

- After the growing phase an error-based pruning is performed.

Random Forest

A Random forest is a classifier consisting of n decision trees. This method combines the *Bagging* (Bootstrap aggregating) method and the random selection of features.

Bagging

Given a data set of n values,

- sample with replacement to get a training set of $m(\leq n)$ values.
- train a decision tree with the m sample values.
- repeat the above procedures until getting t decision trees.

The decision with t decision trees is performed as follows:

- take the majority from the decisions or
- take the average of the decisions

It is known that bagging can alleviate

- underfitting due to high bias as well as
- overfitting due to high variance.

Bagging is a sort of ensemble learning methods that use multiple learning algorithms to obtain better predictive performance.

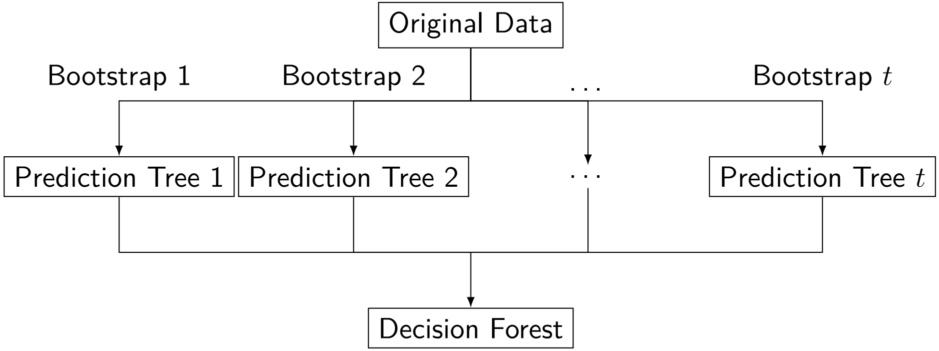


Figure: Bagging

Random Selection of Features

Bagged decision trees have only one parameter, t, the number of decision trees. Random Forests have a second parameter that controls how many features to try when finding the best split.

Suppose there exist m features in the data set. Instead of trying all features every time we make a new decision node, we only try a subset of the features (usually of size $\sqrt{m}$ or $\frac{m}{3}$). This procedure injects randomness into each decision tree, which reduces the correlations among decision trees and hence improves the performance of random forests.

Example 1. Decision Tree

- Iris dataset has four features, sepal length, sepal width, petal length, and petal width, and
- there are 3 classes of iris, setosa, versicolor and virginica.
- We will use 2 features, petal length and petal width, to classify iris dataset by decision tree.

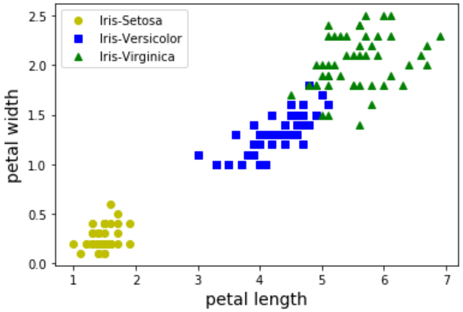
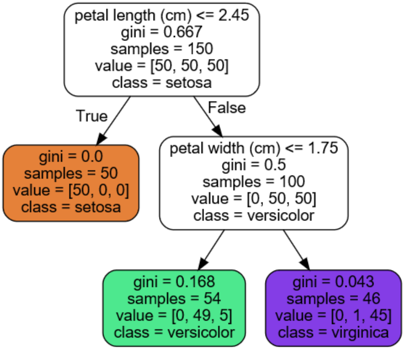


Figure: Iris dataset

Using Gini Index, we can obtain the following decision tree and decision boundaries.



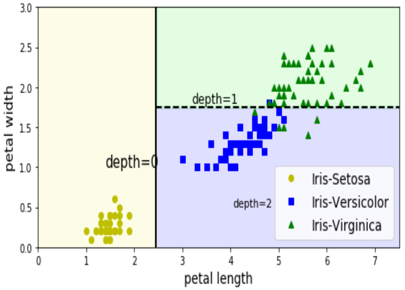
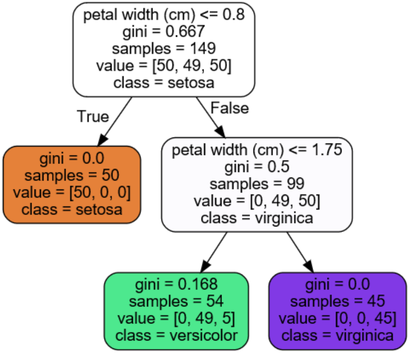


Figure: (a) Decision tree (b) Decision boundary

- Decision tree is sensitive to a little change in data.
- Here we remove one Iris-Versicolor datum with the most wide petal width.



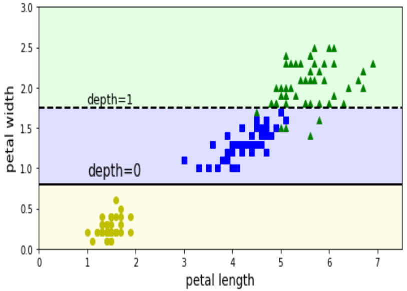


Figure: (a) Decision tree (b) Decision boundary

Example 2. Random Forest

- We generate data by using the `make_moons` function.
- It makes two interleaving half circles.
- We will use decision tree, decision tree using bagging, and random forest for binary classification.

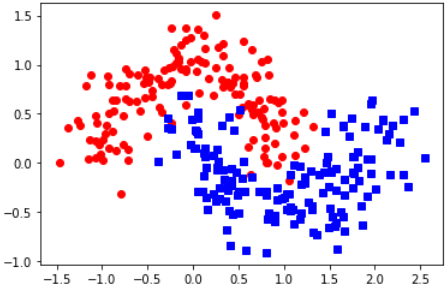


Figure: Randomly generated data $(n = 300)$

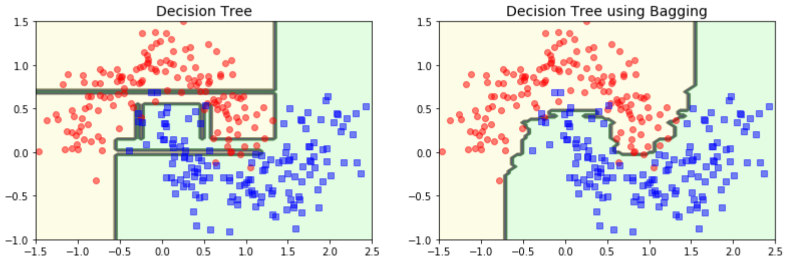


Figure: (a) Decision boundary by decision tree (b) Decision boundary by decision tree using bagging (the number of decision trees: 500, samples: 60)

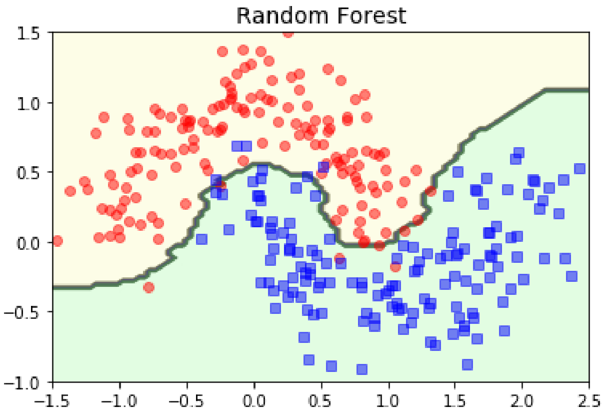


Figure: Decision boundary by random forest (the number of decision trees: 500, samples: 60, sample features: $\sqrt{\text{features}}$)

- In bagging and random forest, decision is made by taking the majority vote.
- In bagging, 500 trees are made. For training each tree, 60 randomly selected samples are used.
- Random forest has one additional process, random feature selection. Among 2 features (x-coordinate and y-coordinate), one feature is randomly selected for each split.

Accuracy is measured by the accuracy score which is the fraction of the correct predictions.

$$\mathrm{accuracy}(y, \hat{y}) := \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\hat{y}_i = y_i)$$

- Decision tree: 0.90666...
- Decision tree using bagging: 0.98666...
- Random forest: 0.96

Example 3. Random Forest

- We generate another set of data from the `make_moons` function.
- It also makes two interleaving half circles.
- We will use decision tree, decision tree using bagging, and random forest for binary classification.

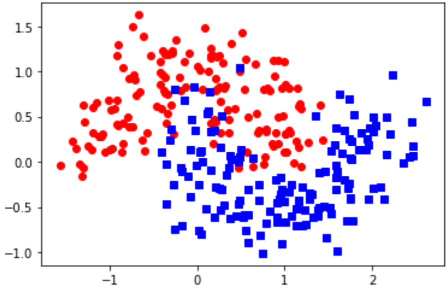
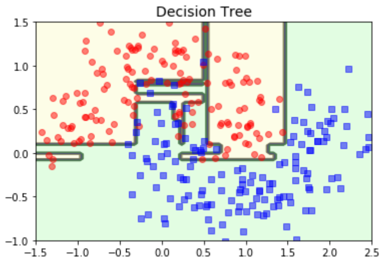


Figure: Randomly generated data ($n = 300$)



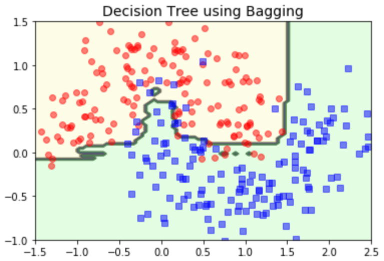


Figure: (a) Decision boundary by decision tree (b) Decision boundary by decision tree using bagging (the number of decision trees: 500, samples: 60)

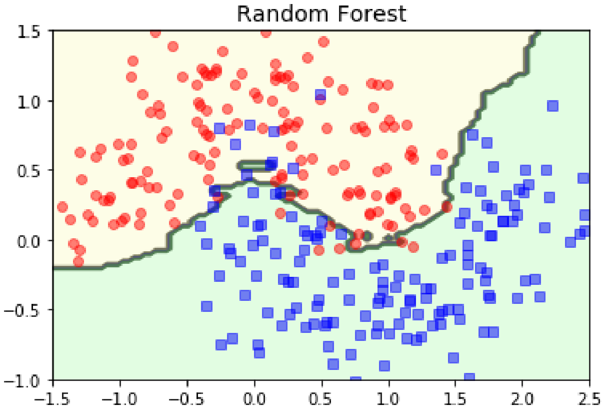


Figure: Decision boundary by random forest (the number of decision trees: 500, samples: 60, sample features: $\sqrt{\text{features}}$)

Accuracy is also measured by the accuracy score.

$$\text{accuracy}(y, \hat{y}) := \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(\hat{y}_i = y_i)$$

- Decision tree: 0.85333...
- Decision tree using bagging: 0.93333...
- Random forest: 0.94666...