MODULE 4

DECISION TREE LEARNING

6.1 Introduction

Decision tree learning model, one of the most popular supervised predictive learning models, classifies data instances with high accuracy and consistency. The model performs an *inductive inference* that reaches a general conclusion from observed examples. This model is variably used for solving complex classification applications.

Decision tree is a concept tree which summarizes the information contained in the training dataset in the form of a tree structure. Once the concept model is built, test data can be easily classified.

- Why called as decision tree ?
  - As starts from root node and finds number of solutions .
- The benefits of having a decision tree are as follows :
  - It does not require any domain knowledge.
  - It is easy to comprehend.
  - The learning and classification steps of a decision tree are simple and fast.
- Example : Toll free number

6.1.1 Structure of a Decision Tree A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

Applies to classification and regression model.

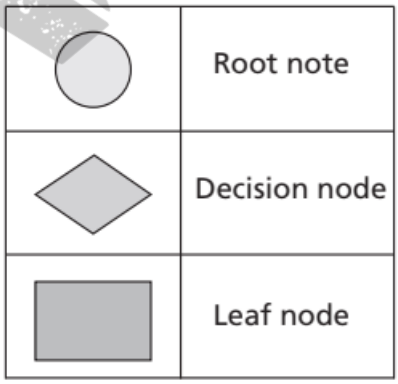

Figure 6.1: Nodes in a Decision Tree

Note: *A decision tree is not always a binary tree. It is a tree which can have more than two branches.*

The decision tree consists of 2 major procedures:

1) Building a tree and

2) Knowledge inference or classification.

Building the Tree

Goal Construct a decision tree with the given training dataset. The tree is constructed in a top-down fashion. It starts from the root node. At every level of tree construction, we need to find the best split attribute or best decision node among all attributes. This process is recursive and continued until we end up in the last level of the tree or finding a leaf node which cannot be split further. The tree construction is complete when all the test conditions lead to a leaf node. The leaf node contains the target class or output of classification.

Output Decision tree representing the complete hypothesis space.

Knowledge Inference or Classification

Goal Given a test instance, infer to the target class it belongs to.

Classification Inferring the target class for the test instance or object is based on inductive inference on the constructed decision tree. In order to classify an object, we need to start traversing the tree from the root. We traverse as we evaluate the test condition on every decision node with the test object attribute value and walk to the branch corresponding to the test's outcome. This process is repeated until we end up in a leaf node which contains the target class of the test object.

Output Target label of the test instance.

Advantages of Decision Trees

1. Easy to model and interpret
2. Simple to understand
3. The input and output attributes can be discrete or continuous predictor variables.
4. Can model a high degree of nonlinearity in the relationship between the target variables and the predictor variables
5. Quick to train

Disadvantages of Decision Trees

Some of the issues that generally arise with a decision tree learning are that:

1. It is difficult to determine how deeply a decision tree can be grown or when to stop growing it.
2. If training data has errors or missing attribute values, then the decision tree constructed may become unstable or biased.
3. If the training data has continuous valued attributes, handling it is computationally complex and has to be discretized.
4. A complex decision tree may also be over-fitting with the training data.
5. Decision tree learning is not well suited for classifying multiple output classes.
6. Learning an optimal decision tree is also known to be NP-complete.

6.1.2 Fundamentals of Entropy

- How to draw a decision tree ?

  Entropy

  Information gain

Entropy is the amount of uncertainty or randomness in the outcome of a random variable or an event. Moreover, entropy describes about the homogeneity of the data instances. The best feature is selected based on the entropy value. For example, when a coin is flipped, head or tail are the two outcomes, hence its entropy is lower when compared to rolling a dice which has got six outcomes.

Higher the entropy → Higher the uncertainty
Lower the entropy → Lower the uncertainty

Let P be the probability distribution of data instances from 1 to n as shown in Eq. (6.2).

So, $P = P_1 \ldots\ldots P_n$ (6.2)

Entropy of P is the information measure of this probability distribution given in Eq. (6.3),

$$\text{Entropy\_Info}(P) = \text{Entropy\_Info}(P_1 \ldots P_n)$$
$$= -(P_1 \log_2(P_1) + P_2 \log_2(P_2) + \ldots\ldots + P_n \log_2(P_n)) \quad (6.3)$$

where, P_1 is the probability of data instances classified as class 1 and P_2 is the probability of data instances classified as class 2 and so on.

P_1 = |No of data instances belonging to class 1| / |Total no of data instances in the training dataset|

Algorithm 6.1: General Algorithm for Decision Trees

Algorithm 6.1: General Algorithm for Decision Trees

1. Find the best attribute from the training dataset using an attribute selection measure and place it at the root of the tree.
2. Split the training dataset into subsets based on the outcomes of the test attribute and each subset in a branch contains the data instances or tuples with the same value for the selected test attribute.
3. Repeat step 1 and step 2 on each subset until we end up in leaf nodes in all the branches of the tree.
4. This splitting process is recursive until the stopping criterion is reached.

Stopping Criteria

The following are some of the common stopping conditions:

1. The data instances are homogenous which means all belong to the same class C_i and hence its entropy is 0.
2. A node with some defined minimum number of data instances becomes a leaf (Number of data instances in a node is between 0.25 and 1.00% of the full training dataset).
3. The maximum tree depth is reached, so further splitting is not done and the node becomes a leaf node.

6.2 DECISION TREE INDUCTION ALGORITHMS

There are many decision tree algorithms, such as ID3, C4.5, CART, CHAID, QUEST, GUIDE, CRUISE, and CTREE, that are used for classification in real-time environment. The most commonly used decision tree algorithms are ID3 (Iterative Dichotomizer 3), developed by J.R Quinlan in 1986, and C4.5 is an advancement of ID3 presented by the same author in 1993. CART, that stands for Classification and Regression Trees, is another algorithm which was developed by Breiman et al. in 1984.

The accuracy of the tree constructed depends upon the selection of the best split attribute. Different algorithms are used for building decision trees which use different measures to decide on the splitting criterion. Algorithms such as ID3, C4.5 and CART are popular algorithms used in the construction of decision trees. The algorithm ID3 uses 'Information Gain' as the splitting criterion whereas the algorithm C4.5 uses 'Gain Ratio' as the splitting criterion. The CART algorithm is popularly used for classifying both categorical and continuous-valued target variables. CART uses GINI Index to construct a decision tree.

6.2.1 ID3 Tree Construction(ID3 stands for Iterative Dichotomiser 3)

A decision tree is one of the most powerful tools of **supervised learning** algorithms used for both **classification** and **regression** tasks.

It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal

node) holds a class label. It is **constructed** by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

Definitions

Let T be the training dataset.
Let A be the set of attributes $A = \{A_1, A_2, A_3, \ldots\ldots A_n\}$.
Let m be the number of classes in the training dataset.
Let P_i be the probability that a data instance or a tuple 'd' belongs to class C_i.
It is calculated as,
P_i = Total no of data instances that belongs to class C_i in T/Total no of tuples in the training set T (6.6)

Mathematically, it is represented as shown in Eq. (6.7).

$$P_i = \frac{|d_{C_i}|}{|T|} \quad (6.7)$$

$$\text{Entropy\_Info}(T) = -\sum_{i=1}^{m} P_i \log_2 P_i \quad (6.8)$$

$$\text{Entropy\_Info}(T, A) = \sum_{i=1}^{v} \frac{|A_i|}{|T|} \times \text{Entropy\_Info}(A_i) \quad (6.9)$$

$$\text{Information\_Gain}(A) = \text{Entropy\_Info}(T) - \text{Entropy\_Info}(T, A) \quad (6.10)$$

Algorithm 6.2: Procedure to Construct a Decision Tree using ID3

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9) and Information_Gain Eq. (6.10) for each of the attribute in the training dataset.
3. Choose the attribute for which entropy is minimum and therefore the gain is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

6.2.2 C4.5 Construction

C4.5 is a widely used algorithm for constructing decision trees from a dataset.

Disadvantages of ID3 are: Attributes must be nominal values, dataset must not include missing data, and finally the algorithm tend to fall into overfitting.

To overcome this disadvantage Ross Quinlan, inventor of ID3, made some improvements for these bottlenecks and created a new algorithm named C4.5. Now, the algorithm can create a more generalized models including continuous data and could handle missing data. And also works with discrete data, supports post-prunning.

Given a Training dataset T,

The Split_Info of an attribute A is computed as given in Eq. (6.11):

$$\text{Split\_Info}(T, A) = -\sum_{i=1}^{v} \frac{|A_i|}{|T|} \times \log_2 \frac{|A_i|}{|T|}$$

where, the attribute A has got 'v' distinct values $\{a_1, a_2, \ldots a_v\}$, and $|A_i|$ is the number of instances for distinct value 'i' in attribute A.

The Gain_Ratio of an attribute A is computed as given in Eq. (6.12):

$$\text{Gain\_Ratio}(A) = \frac{\text{Info\_Gain}(A)}{\text{Split\_Info}(T, A)}$$

Algorithm 6.3: Procedure to Construct a Decision Tree using C4.5

1. Compute Entropy_Info Eq. (6.8) for the whole training dataset based on the target attribute.
2. Compute Entropy_Info Eq. (6.9), Info_Gain Eq. (6.10), Split_Info Eq. (6.11) and Gain_Ratio Eq. (6.12) for each of the attribute in the training dataset.
3. Choose the attribute for which Gain_Ratio is maximum as the best split attribute.
4. The best split attribute is placed as the root node.
5. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into subsets.
6. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

Dealing with Continuous Attributes in C4.5

The C4.5 algorithm is further improved by considering attributes which are continuous, and a continuous attribute is discretized by finding a split point or threshold. When an attribute '*A*' has numerical values which are continuous, a threshold or best split point '*s*' is found such that the set of values is categorized into two sets such as $A < s$ and $A \geq s$. The best split point is the attribute value which has maximum information gain for that attribute.

Now, let us consider the set of continuous values for the attribute CGPA in the sample dataset as shown in Table 6.12.

Table 6.12: Sample Dataset

S.No.	CGPA	Job Offer
1.	9.5	Yes
2.	8.2	Yes
3.	9.1	No
4.	6.8	No
5.	8.5	Yes
6.	9.5	Yes
7.	7.9	No
8.	9.1	Yes
9.	8.8	Yes
10.	8.8	Yes

First, sort the values in an ascending order.

6.8	7.9	8.2	8.5	8.8	8.8	9.1	9.1	9.5	9.5

Remove the duplicates and consider only the unique values of the attribute.

6.8	7.9	8.2	8.5	8.8	8.8	9.1	9.5

Now, compute the Gain for the distinct values of this continuous attribute. Table 6.13 shows the computed values.

Table 6.13: Gain Values for CGPA

	6.8		7.9		8.2		8.5		8.8		9.1		9.5	
Range	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>	≤	>
Yes	0	7	0	7	1	6	2	5	4	3	5	2	7	0
No	1	2	2	1	2	1	2	1	2	1	3	0	3	0
Entropy	0	0.7637	0	0.5433	0.9177	0.5913	1	0.6497	0.9177	0.8108	0.9538	0	0.8808	0
Entropy_Info (*S*, *T*)	0.6873		0.4346		0.6892		0.7898		0.8749		0.7630		0.8808	
Gain	0.1935		**0.4462**		0.1916		0.091		0.0059		0.1178		0	

Table 6.14: Discretized Instances

S.No.	CGPA Continuous	CGPA Discretized	Job Offer
1.	9.5	>7.9	Yes
2.	8.2	>7.9	Yes
3.	9.1	>7.9	No
4.	6.8	≤7.9	No
5.	8.5	>7.9	Yes
6.	9.5	>7.9	Yes
7.	7.9	≤7.9	No
8.	9.1	>7.9	Yes
9.	8.8	>7.9	Yes
10.	8.8	>7.9	Yes

6.2.3 Classification and Regression Trees Construction

Classification and Regression Trees (CART) is a widely used algorithm for constructing decision trees that can be applied to both classification and regression tasks. CART is similar to C4.5 but has some differences in its construction and splitting criteria.

The classification method CART is required to construct a decision tree based on Gini's impurity index. It serves as an example of how the values of other variables can be used to predict the values of a target variable. It functions as a fundamental machine-learning method and provides a wide range of use cases

subset. For example, if an attribute A has three distinct values say $\{a_1, a_2, a_3\}$, the possible subsets are $\{\ \}$, $\{a_1\}$, $\{a_2\}$, $\{a_3\}$, $\{a_1, a_2\}$, $\{a_1, a_3\}$, $\{a_2, a_3\}$, and $\{a_1, a_2, a_3\}$. So, if an attribute has 3 distinct values, the number of possible subsets is 2^3, which means 8. Excluding the empty set $\{\ \}$ and the full set $\{a_1, a_2, a_3\}$, we have 6 subsets. With 6 subsets, we can form three possible combinations such as:

$\{a_1\}$ with $\{a_2, a_3\}$

$\{a_2\}$ with $\{a_1, a_3\}$

$\{a_3\}$ with $\{a_1, a_2\}$

Hence, in this CART algorithm, we need to compute the best splitting attribute and the best split subset i in the chosen attribute.

Higher the GINI value, higher is the homogeneity of the data instances.

Gini_Index(T) is computed as given in Eq. (6.13).

$$\text{Gini\_Index}(T) = 1 - \sum_{i=1}^{w} p_i^2 \qquad (6.13)$$

where,

P_i be the probability that a data instance or a tuple 'd' belongs to class C_i. It is computed as:

P_i = |No. of data instances belonging to class i|/|Total no of data instances in the training dataset T|

GINI Index assumes a binary split on each attribute, therefore, every attribute is considered as a binary attribute which splits the data instances into two subsets S_1 and S_2.

Gini_Index(T, A) is computed as given in Eq. (6.14).

$$\text{Gini\_Index}(T, A) = \frac{|S_1|}{|T|}\text{Gini}(S_1) + \frac{|S_2|}{|T|}\text{Gini}(S_2) \quad (6.14)$$

The splitting subset with minimum Gini_Index is chosen as the best splitting subset for an attribute. The best splitting attribute is chosen by the minimum Gini_Index which is otherwise maximum ΔGini because it reduces the impurity.

ΔGini is computed as given in Eq. (6.15):

$$\Delta\text{Gini}(A) = \text{Gini}(T) - \text{Gini}(T, A) \quad (6.15)$$

Algorithm 6.4: Procedure to Construct a Decision Tree using CART

1. Compute Gini_Index Eq. (6.13) for the whole training dataset based on the target attribute.
2. Compute Gini_Index for each of the attribute Eq. (6.14) and for the subsets of each attribute in the training dataset.
3. Choose the best splitting subset which has minimum Gini_Index for an attribute.
4. Compute ΔGini Eq. (6.15) for the best splitting subset of that attribute.
5. Choose the best splitting attribute that has maximum ΔGini.
6. The best split attribute with the best split subset is placed as the root node.
7. The root node is branched into two subtrees with each subtree an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into two subsets.
8. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

6.2.4 Regression Trees

Regression trees are a variant of decision trees where the target feature is a continuous valued variable. These trees can be constructed using an algorithm called reduction in variance which uses standard deviation to choose the best splitting attribute.

Algorithm 6.5: Procedure for Constructing Regression Trees

1. Compute standard deviation for each attribute with respect to target attribute.
2. Compute standard deviation for the number of data instances of each distinct value of an attribute.
3. Compute weighted standard deviation for each attribute.

4. Compute standard deviation reduction by subtracting weighted standard deviation for each attribute from standard deviation of each attribute.
5. Choose the attribute with a higher standard deviation reduction as the best split attribute.
6. The best split attribute is placed as the root node.
7. The root node is branched into subtrees with each subtree as an outcome of the test condition of the root node attribute. Accordingly, the training dataset is also split into different subsets.
8. Recursively apply the same operation for the subset of the training set with the remaining attributes until a leaf node is derived or no more training instances are available in the subset.

6.3 VALIDATING AND PRUNING OF DECISION TREES

Validating and pruning decision trees is a crucial part of building accurate and robust machine learning models. Decision trees are prone to overfitting, which means they can learn to capture noise and details in the training data that do not generalize well to new, unseen data.

Validation and pruning are techniques used to mitigate this issue and improve the performance of decision tree models.

The **pre-pruning** technique of Decision Trees is tuning the hyperparameters prior to the training pipeline. It involves the heuristic known as 'early stopping' which stops the growth of the decision tree - preventing it from reaching its full depth. It stops the tree-building process to avoid producing leaves with small samples. During each stage of the splitting of the tree, the cross-validation error will be monitored. If the value of the error does not decrease anymore - then we stop the growth of the decision tree.

The hyperparameters that can be tuned for early stopping and preventing overfitting are: max_depth, min_samples_leaf, and min_samples_split

These same parameters can also be used to tune to get a robust model

Post-pruning does the opposite of pre-pruning and allows the Decision Tree model to grow to its full depth. Once the model grows to its full depth, tree branches are removed

to prevent the model from overfitting. The algorithm will continue to partition data into smaller subsets until the final subsets produced are similar in terms of the outcome variable. The final subset of the tree will consist of only a few data points allowing the tree to have learned the data to the T. However, when a new data point is introduced that differs from the learned data - it may not get predicted well.

The hyperparameter that can be tuned for post-pruning and preventing overfitting is: ccp_alpha

ccp stands for Cost Complexity Pruning and can be used as another option to control the size of a tree. A higher value of ccp_alpha will lead to an increase in the number of nodes pruned.

Some of the tree pruning methods are listed below:

1. Reduced Error Pruning
2. Minimum Error Pruning (MEP)
3. Pessimistic Pruning
4. Error–based Pruning (EBP)
5. Optimal Pruning
6. Minimum Description Length (MDL) Pruning
7. Minimum Message Length Pruning
8. Critical Value Pruning