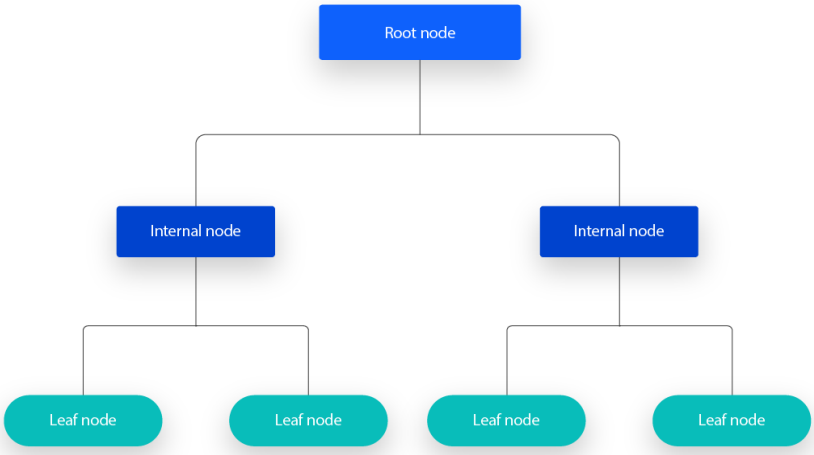
Machine Learning Lecture 3 (Decision Trees)

By Mohamed Hafez

Decision Tree

A decision tree is a tree-like structure that helps make decisions or predictions based on certain conditions or features. Think of it like a flowchart with different paths, where each node represents a feature or attribute, and each branch represents a possible outcome or decision based on that feature. Decision trees are commonly used in machine learning for tasks such as classification and regression. By following the branches of the tree, we can determine the final decision or prediction.



Decision Tree Intuition

- The intuition behind a decision tree is quite straightforward. Imagine playing a game of 20 questions, where you have to guess an object by asking yes-or-no questions. A decision tree works similarly. It starts with a root node that represents the starting question or condition. Based on the answer, you move down the tree to a child node that represents the next question or condition.

- The goal of a decision tree is to ask questions or test conditions that efficiently split the data into smaller and more homogeneous groups. This process continues recursively, splitting the data until reaching leaf nodes, which represent the final decisions or predictions.

- The intuition is that decision trees learn by finding the most effective questions or conditions that maximize the separation or information gain in the data. By making decisions based on these questions, decision trees can predict or classify new instances with a high degree of accuracy.

Classification trees

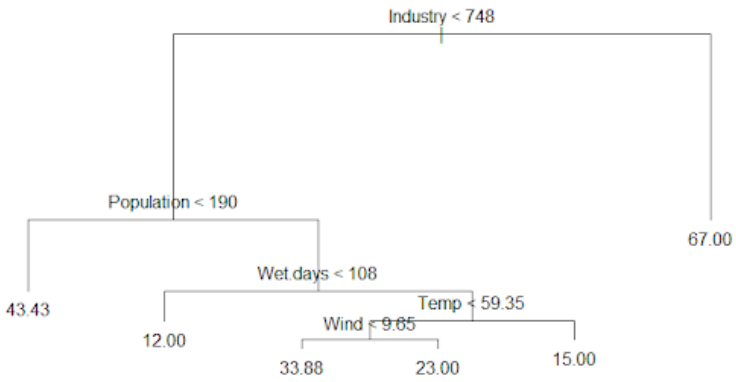
- A classification tree is a type of decision tree that is used to classify or categorize data into different groups or classes.

- Think of a classification tree as a flowchart where each node represents a question or condition, and each branch represents a possible outcome or class. The goal of a classification tree is to find the best questions or conditions that split the data into distinct classes, making accurate predictions for new or unseen data.

- Classification trees are commonly used in machine learning to solve classification problems, such as identifying whether an email is spam or not, predicting the species of a flower based on its features, or classifying images into different categories. By following the branches of the classification tree, the algorithm determines the class to which a new data point belongs.

Regression Trees

- a regression tree is another type of decision tree used in machine learning. While classification trees are used for predicting discrete classes or categories, regression trees are used for predicting continuous numeric values.



- Let's consider an example for better understanding. Imagine we want to predict the price of a house based on its features like location, size, number of rooms, etc. In this case, we have a regression problem because we want to predict a numeric value (price). A regression tree can help with that.

- The regression tree starts with a root node that represents a question or condition related to the features. For instance, it may ask whether the house is in a certain neighbourhood.

Based on the answer (e.g., yes or no), the tree branches out to child nodes representing subsequent questions or conditions.

- Each node in the regression tree aims to create splits based on the feature conditions that minimize the differences or errors in predicting the target value. The goal is to find the best questions to divide the data such that the predicted values within each split are as close as possible to the actual values.

- By following the branches, we reach the leaf nodes, which contain the predicted values. In the case of house price prediction, each leaf node might represent a specific price range.

- To summarize, a regression tree helps us predict continuous values by dividing the data based on different conditions, and ultimately providing predictions at the leaf nodes based on the conditions followed.

CART

- CART stands for Classification And Regression Trees. In very simple words, CART is a type of decision tree algorithm that can be used for both classification and regression tasks. Watch this video for further explanation:
https://www.youtube.com/watch?app=desktop&v=nnUHcLZ8Q_k

- Let's see how CART works with an example. Suppose we have a dataset with information about different types of fruits, including their color, size, and taste. If our goal is to predict the type of fruit (classification), we can use CART to build a classification tree.

- CART starts with the entire dataset at the root node and selects the best feature to split the data based on criteria like entropy or Gini impurity. For example, it might split the data based on the color of the fruit. The tree then creates child nodes representing different colors like red, green, or yellow.

- The splitting continues recursively, considering other attributes such as size or taste, until it reaches leaf nodes that represent the predicted fruit type (e.g., apple, orange, etc.). This allows us to classify new fruits by following the branches of the tree based on their color, size, and taste.

- Alternatively, if our goal is to predict a continuous value like the price of a fruit (regression), we can use CART for regression tasks. Instead of splitting based on categorial features like

color, it would split the data based on numerical features like size or weight. The leaf nodes in this case would contain predicted values for the fruit's price based on the given attributes.

- In summary, CART is a versatile decision tree algorithm that can be used for both classification and regression tasks. It helps us make predictions by recursively splitting the data based on features until reaching leaf nodes that represent class labels or predicted values.

Non-parametric estimation & non-parametric model

- Non-parametric estimation and non-parametric models are approaches in statistics and machine learning that do not make specific assumptions about the shape or form of the underlying data distribution. Instead, they aim to estimate the relationship between variables or make predictions without assuming a particular mathematical formula.

- Let's look at some simple examples to understand non-parametric estimation and modeling better:

- Non-parametric estimation: Suppose we want to estimate the average salary of employees in a company. In non-parametric estimation, we don't make assumptions about the distribution of salaries. Instead, we can use a method called the sample median, which estimates the middle value of the salary data. This method does not assume any particular form for the data and provides a robust estimate of the central tendency.
- Non-parametric model: Consider a scenario where we want to predict the price of a house based on its features such as size, number of rooms, and location. Instead of assuming a specific mathematical relationship like linear regression, we can use a non-parametric model called random forest. Random forest builds an ensemble of decision trees, allowing for complex interactions and nonlinearities between the features. This model does not impose strong assumptions on the relationship between the features and the target variable.

- In both cases, non-parametric estimation and modeling offer flexibility by avoiding assumptions about the form or distribution of the data. They can handle diverse patterns and allow for more complex relationships between variables.

- In simple terms, non-parametric estimation is about estimating values or characteristics of data without assuming a specific distribution, while non-parametric models are flexible models that can capture complex relationships without relying on specific mathematical formulas.

Top-down learning

- Top-down learning, also known as deductive learning, is an approach where we start with general rules or theories and use them to understand specific examples or make predictions.

It involves reasoning from the top, or the general principles, to the bottom, which is the specific details.

- In top-down learning, we begin with a broad understanding or knowledge that is already established or given. We then apply this knowledge to specific situations or examples to derive conclusions or make predictions. It is a deductive approach because it involves using existing information or theories to draw logical inferences.

- For example, let's say we are learning about different animal classifications. In top-down learning, we would start with the broad categories such as mammals, reptiles, and birds. We then apply these general rules or classifications to specific examples, such as identifying a dog as a mammal because it has certain characteristics like fur, giving birth to live young, and being warm-blooded.

- In simple terms, top-down learning is an approach where we use general knowledge or rules to understand specific examples or make predictions. It involves reasoning from the top, or the general principles, to the bottom, which is the specific details.

ID3 Algorithm

- The ID3 algorithm is a popular algorithm used in machine learning for building decision trees. In simple terms, it is a method for automatically creating a decision tree from a given dataset.
- Here's how the ID3 algorithm works with an example:
- Suppose we have a dataset with information about whether or not to play tennis based on weather conditions like outlook, temperature, humidity, and wind. The goal is to build a decision tree that can predict whether or not to play tennis based on these weather conditions.

1. The ID3 algorithm starts with the entire dataset at the root node of the decision tree.
2. It evaluates different attributes (in this case, outlook, temperature, humidity, and wind) to determine the one that provides the most useful or informative splitting of the data. This is typically done using a metric like information gain.
3. The attribute with the highest information gain is selected as the splitting criterion or question at the current node. For example, if outlook provides the most information gain, it would be chosen as the question at the root node.
4. The algorithm creates child nodes for each possible value of the chosen attribute. For example, if the values of outlook are "sunny," "overcast," and "rainy," three child nodes would be created.

5. The algorithm recursively applies the same process to each child node, considering the remaining attributes and selecting the one with the highest information gain as the next splitting criterion.
6. This process continues until a stopping condition is met, such as when all instances at a node have the same class label, or when there are no more attributes to split on.

The result is a decision tree that can be used to predict whether or not to play tennis based on the weather conditions.

Information gain

- Information gain is a concept used in decision tree algorithms, such as the ID3 algorithm, to measure how much information a particular attribute provides in terms of splitting the data and making predictions.

- In very simple terms, information gain tells us how much a feature or attribute reduces the uncertainty or randomness in the data when we consider it for splitting.

- Here's an example to help illustrate information gain:

- Imagine we have a dataset with information about whether or not to play tennis based on weather conditions. We want to build a decision tree to predict whether tennis will be played or not. One of the attributes is "outlook," which can take values like "sunny," "overcast," or "rainy."

To calculate the information gain for the "outlook" attribute, we consider the following steps:

1. Calculate the initial level of uncertainty or randomness in the data. This is typically done using a metric called entropy.
2. Split the data based on the possible values of the "outlook" attribute. For example, we divide the dataset into subsets of instances where the outlook is sunny, overcast, or rainy.
3. Calculate the entropy of each subset. Entropy measures the impurity or randomness in a given set of data.
4. Calculate the weighted average of the entropies of the subsets, taking into account the proportion of instances in each subset. This gives us the overall entropy after the split.
5. Finally, calculate the information gain by subtracting the overall entropy after the split from the initial entropy. The higher the information gain, the more informative the attribute is for splitting the data.

- In our example, if the information gain for the "outlook" attribute is high, it means that it significantly reduces the uncertainty in the data when used for splitting. This indicates that the "outlook" attribute is informative in predicting whether tennis will be played or not based on the weather conditions.

- So, in simple terms, information gain measures how much an attribute reduces the uncertainty or randomness in the data when used for splitting in a decision tree algorithm. It helps determine which attribute is the most informative for making accurate predictions.

Impurity & Entropy

- In the context of decision trees and machine learning, impurity and entropy are concepts used to measure the randomness or unpredictability of a set of data.

- Impurity refers to the degree of mixed or heterogeneous class labels within a dataset. When a dataset has a high level of impurity, it means that the classes or categories within the data are mixed or not well-separated.

- Entropy, on the other hand, is a specific metric used to measure impurity. It quantifies the amount of uncertainty in a dataset or a given set of data. A higher entropy value indicates higher unpredictability or randomness, while a lower entropy value indicates more certainty or order.

- Here's an example to help understand impurity and entropy:

- Suppose we have a dataset with information about whether or not to play tennis based on weather conditions. The dataset contains instances where the weather can be either "sunny," "overcast," or "rainy," and the target variable is the decision to play tennis or not.

- To calculate the entropy of this dataset, we consider the class distribution or the proportions of instances belonging to each class.

- If our dataset has equal numbers of instances playing tennis and not playing tennis, the distribution is perfectly balanced. In this case, the entropy is at its highest because the class labels are mixed and unpredictable.

- However, if the dataset contains only instances where everyone plays tennis, the entropy is at its lowest because the class labels are perfectly predictable and not mixed.

Impurity criterion (Gini index & entropy)

- In decision tree algorithms, impurity criterion, Gini index, and entropy are methods used to measure the impurity or uncertainty in a set of data. They help determine the best split for creating decision trees.

Impurity Criterion

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

- Impurity Criterion: It is a general term that refers to the measure used to quantify the impurity of a set of data. Examples of impurity criteria include Gini index and entropy.

- Gini Index: The Gini index is a measure of impurity often used in classification tasks. It calculates the probability that a randomly selected element from a dataset would be incorrectly labeled if it were randomly labeled according to the distribution of class labels in the subset. The Gini index ranges from 0 to 1, where 0 represents perfect purity (all elements belong to a single class) and 1 represents maximum impurity (an equal distribution of classes).

- For example, consider a dataset with two classes, "spam" and "not spam." If the dataset is perfectly divided, where all instances are labeled as either "spam" or "not spam," the Gini index would be 0. If the classes are evenly mixed in the dataset, the Gini index would be 0.5.

- Entropy: Entropy is another measure of impurity used in decision trees. It calculates the disorder or randomness within a set of data. The entropy formula calculates the probability of occurrence for each class label and sums the products of those probabilities with their logarithms. Higher entropy values represent higher disorder and uncertainty in the data, while lower entropy values represent more order and predictability.

- For example, suppose we have a dataset with three classes: "cat," "dog," and "bird." If the dataset consists of an equal number of instances for each class, the entropy would be at its

highest, indicating maximum disorder. In contrast, if all instances in the dataset belong to the same class, the entropy would be at its lowest, indicating perfect order and predictability.

Class cases in entropy

- When all the examples in a group belong to the same class, the entropy is low because there is no uncertainty or disorder. It's like having a group of only cats, with no dogs or any other animals. The entropy is low because we are certain that all the examples belong to the same class.

- On the other hand, when a group has an equal number of examples from each class (50% in either class), the entropy is high because there is more uncertainty or disorder. It's like having a group with an equal number of cats and dogs. Since we are unsure about the class of each example, the entropy is high.

- So, in simple terms, the entropy of a group is low when all the examples belong to the same class, and high when there is an equal number of examples from each class.

Specific conditional entropy, conditional entropy, and Mutual Information

- Specific Conditional Entropy: Conditional entropy measures the uncertainty in a random variable given some additional information. Specific conditional entropy refers to the uncertainty of a random variable given a specific value of another variable. Let's say you have a box of balls and you want to know the entropy of the colors of the balls given that you already know the color of one ball is red. The specific conditional entropy would measure how uncertain or unpredictable the remaining balls' colors are when one ball is already known to be red.

- Conditional Entropy: Conditional entropy, on the other hand, calculates the average uncertainty of a random variable using the probability of the other variable's values. If you have a box of balls with different colors, and you want to calculate the average uncertainty of the colors of the balls based on the probability distribution of another variable, you would use conditional entropy. It measures the amount of uncertainty in a random variable after considering the values of another variable.

- Mutual Information: Mutual information measures the amount of information that two random variables share. It quantifies how much one variable tells you about the other. Using the balls example, mutual information would measure how much information knowing the color of one ball gives you about the color of another ball.

- In summary, entropy measures uncertainty or unpredictability of a random variable, specific conditional entropy measures uncertainty given a specific value of another variable, conditional entropy measures average uncertainty given the probability distribution of another variable, and mutual information measures how much information two random variables share.

Calculating information gain

- Imagine you are playing a guessing game where you need to guess the color of a ball from a box. The box contains 100 balls, and the balls can be either red or blue. You are allowed to ask yes-or-no questions about the ball to guess its color correctly.

- Now, let's say you initially have no information about the ball and its color. The entropy (uncertainty) of the box is high because you have no clue about the color distribution.

- To make an informed guess, you decide to ask a question: "Is the ball red?" By asking this question, you gain some information. The information gain is calculated by comparing the entropy before and after asking the question.

- If, let's say, the box contains 80 red balls and 20 blue balls, asking the question "Is the ball red?" would result in high information gain. This is because the question helps you eliminate a large number of possibilities (80 red balls) and reduces the uncertainty significantly.

- On the other hand, if the box contains an equal number of red and blue balls (50 red balls and 50 blue balls), asking the question "Is the ball red?" would result in lower information gain. This is because both possibilities are equally likely, and the question doesn't provide much information to narrow down the options significantly.

- In simple terms, information gain is a measure of how much a question reduces the uncertainty or entropy of the situation. The higher the information gain, the more valuable the question is in helping us make an informed guess or decision.

Overfitting in Decision Trees (And how to avoid it)

- Imagine you want to build a decision tree to predict whether a person will like a movie or not based on their age and genre preference. You have a dataset of movie ratings from different people, including their age, genre preference, and whether they liked the movie or not.

- When building a decision tree, the goal is to create a tree that can accurately predict whether a new person will like a movie or not based on their age and genre preference.

- However, overfitting can occur when a decision tree becomes overly complex and captures too much noise or random patterns in the training data. This leads to poor generalization on new, unseen data.

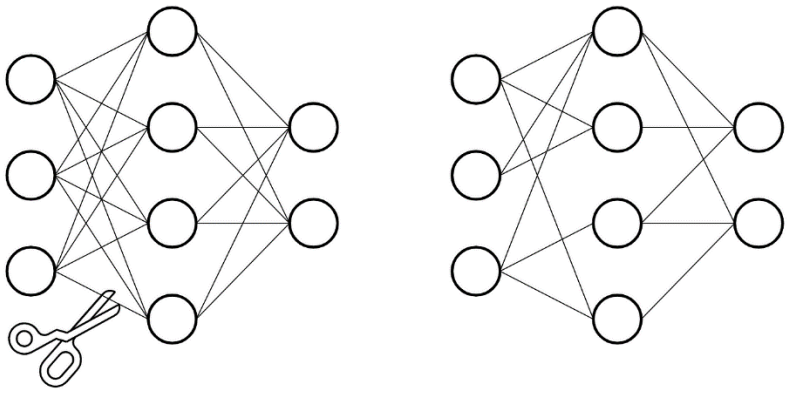
- For example, let's say you have a decision tree with many branches and splits that perfectly fit the training data. It accurately captures every single rating in the dataset. This is a sign of overfitting because the decision tree has become too specific to the training data, and it may not generalize well to new situations.

To avoid overfitting in decision trees, you can take the following steps:

1. Limit the depth of the tree: Instead of allowing the decision tree to continue growing until it perfectly fits the training data, limit the depth of the tree. This helps prevent the tree from becoming too complex and overfitting the training data.
2. Use pruning techniques: Pruning involves reducing the size of a decision tree by removing branches that provide little additional information. This helps simplify the tree and prevent overfitting.
3. Increase the minimum number of samples per leaf: Setting a higher threshold for the minimum number of samples required in each leaf node can prevent the decision tree from creating very small branches that capture noise in the data.
4. Cross-validation: Splitting the data into training and validation sets and using cross-validation techniques can help evaluate the performance of the decision tree on unseen data. This allows you to choose the optimal tree that balances accuracy and generalization.

Reduced error pruning

- Reduced error pruning is a technique used to simplify or reduce the complexity of a decision tree by removing branches or sub-trees that do not significantly improve the tree's accuracy.



Before pruning

After pruning

- Imagine you have a decision tree that predicts whether a person will enjoy a specific type of music based on their age and favorite color. The decision tree has many branches and splits that accurately capture the patterns in the training data.
- In reduced error pruning, you evaluate the performance of the decision tree on a separate validation dataset. You traverse the tree and evaluate the accuracy on the validation data at each node or sub-tree.

- Starting from the bottom of the tree, you consider removing individual branches or entire sub-trees and evaluate the impact on accuracy. If removing a branch or sub-tree does not significantly reduce the accuracy on the validation data, it is pruned or removed from the tree.

- The goal is to simplify the decision tree by removing unnecessary branches or sub-trees that do not contribute much to improving accuracy on new, unseen data. By doing this, reduced error pruning helps prevent overfitting, where the decision tree becomes too specific to the training data and performs poorly on new data.

- For example, suppose you remove a branch or sub-tree that only has a small number of examples but does not improve accuracy significantly. This pruning step simplifies the decision tree, making it more general and less prone to overfitting.

- Reduced error pruning strikes a balance between accuracy and simplicity, resulting in a decision tree that generalizes well to new data while still maintaining reasonable accuracy.

Where should we use decision trees?

Decision trees are a useful tool in various problem-solving scenarios where we need to make decisions based on multiple factors or features. Here are some examples of where decision trees can be used:

1. Classification Problems: Decision trees can be used for classifying data into different categories or classes. For example, determining whether an email is spam or not, classifying images into different object categories, or predicting whether a customer will buy a product or not.
2. Regression Problems: Decision trees can also be used for predicting numeric or continuous values. For example, predicting the price of a house based on its features such as size, location, and number of rooms.
3. Feature Selection: Decision trees can help identify important features or variables that contribute the most to a particular outcome. The tree can provide insights into which features are more influential in making a decision.

4. Exploratory Analysis: Decision trees can aid in exploring and understanding data by visualizing the relationships between different variables. They provide a clear and intuitive representation of how decisions are made based on various conditions.
5. Risk Assessment: Decision trees can be used in risk assessment and decision-making processes, such as determining whether to approve a loan application based on factors like income, credit history, and employment status.
6. Medical Diagnosis: Decision trees can assist in medical diagnosis by considering various symptoms, test results, and medical history to classify patients into different disease categories or recommend appropriate treatments.