Decision Tree

Luu Minh Sao Khue

December 9, 2025

1. Introduction

Decision Trees are simple yet powerful predictive models that learn a sequence of *if–then* rules from data. They can be used for both **classification** (predicting a discrete label) and **regression** (predicting a numerical value).

Key characteristics:

- The model has a tree structure: internal nodes are tests on features, and leaves store predictions.
- The input space is recursively partitioned into regions with increasingly homogeneous targets.
- The same algorithmic idea works for both regression and classification, only the *impurity measure* (loss at a node) changes.
- Decision Trees are interpretable: each path from the root to a leaf can be read as a transparent decision rule.

Decision Trees are base learners for many ensemble methods such as Random Forests, Gradient Boosting Trees, XGBoost, LightGBM, and CatBoost.

2. Intuition

Suppose we have training data

$$\{(x_i, y_i)\}_{i=1}^{N},$$

where $x_i \in \mathbb{R}^d$ is a feature vector and y_i is either a class label (classification) or a real number (regression).

The idea of a decision tree:

> *Recursively split the input space into smaller regions so that within each region the target values y_i are as "pure" or homogeneous as possible.*

At each internal node:

- We choose a feature j and a threshold t (for numerical features) or subset of categories (for categorical features).
- We split the data into left and right child nodes according to this test.
- We want this split to *reduce impurity* as much as possible.

At each leaf:

- In classification, we predict the majority class in that leaf.
- In regression, we predict the average of the target values in that leaf.

The algorithm continues splitting until some stopping condition is reached (e.g. maximum depth, minimum number of samples in a leaf, or no further impurity reduction).

3. Node Impurity and Splitting Criteria

The central mathematical idea in decision trees is the notion of **impurity** at a node. A node is pure if all samples at that node have the same class (classification) or very similar target values (regression). A good split is one that *reduces* impurity.

Let R be a node containing a subset of training samples

$$R = \{(x_i, y_i) : i \in I_R\},$$

where I_R is the index set of samples reaching node R.

We denote the number of samples in R by

$$N_R = |I_R|.$$

3.1 Classification: Class Probabilities and Gini Impurity

Assume we have K classes:

$$\mathcal{Y} = \{1, 2, \ldots, K\}.$$

At node R, we estimate the class probabilities by

$$\hat{p}_{kR} = \frac{1}{N_R} \sum_{i \in I_R} \mathbb{I}(y_i = k), \quad k = 1, \ldots, K.$$

Here:

- $\mathbb{I}(\cdot)$ is the indicator function (1 if the condition is true, 0 otherwise).
- $\hat{p}_{kR}$ is the empirical probability of class k in node R.

The **Gini impurity** at node R is defined as

$$G(R) = \sum_{k=1}^{K} \hat{p}_{kR}\big(1 - \hat{p}_{kR}\big) = 1 - \sum_{k=1}^{K} \hat{p}_{kR}^2.$$

Interpretation.

- If all samples belong to a single class, say class $k^\star$, then $\hat{p}_{k^\star R} = 1$ and all other $\hat{p}_{kR} = 0$. Hence,
$$G(R) = 1 - 1^2 = 0,$$
which corresponds to a perfectly pure node.
- If the classes are uniformly distributed, i.e. $\hat{p}_{kR} = 1/K$ for all k, then
$$G(R) = 1 - K\left(\frac{1}{K}\right)^2 = 1 - \frac{1}{K},$$
which is the maximum impurity for K classes.

Thus $G(R)$ measures how mixed the node is: small $G(R)$ means high purity; large $G(R)$ means low purity.

3.2 Classification: Entropy and Information Gain

Another impurity measure is the **entropy** at node R:

$$H(R) = -\sum_{k=1}^{K} \hat{p}_{kR} \log \hat{p}_{kR},$$

where by convention $0 \log 0$ is taken to be 0.

Properties.

- $H(R) \geq 0$.
- $H(R) = 0$ if and only if the node is pure (all samples have the same class).
- $H(R)$ is largest when the classes are equally likely.

When we consider a split of node R into two child nodes R_{left} and R_{right}, the entropy after the split is
$$H_{\text{split}} = \frac{N_{\text{left}}}{N_R} H(R_{\text{left}}) + \frac{N_{\text{right}}}{N_R} H(R_{\text{right}}),$$
where N_{left} and N_{right} are the number of samples in the left and right child nodes.

The **information gain** of the split is

$$\text{IG} = H(R) - H_{\text{split}}.$$

A good split maximizes this information gain (equivalently, minimizes the weighted entropy after the split).

3.3 Regression: Mean Squared Error (MSE) Impurity

For regression, the target y is real-valued. At node R, we define the average target value

$$\bar{y}_R = \frac{1}{N_R} \sum_{i \in I_R} y_i.$$

The natural impurity measure is the sum of squared errors:

$$\mathrm{SSE}(R) = \sum_{i \in I_R} (y_i - \bar{y}_R)^2.$$

Sometimes we use the mean squared error (MSE),

$$\mathrm{MSE}(R) = \frac{1}{N_R} \sum_{i \in I_R} (y_i - \bar{y}_R)^2,$$

but for comparing splits, the factor $1/N_R$ is constant, so we can equivalently work with $\mathrm{SSE}(R)$.

When splitting node R into R_{left} and R_{right}, the total squared error after the split is

$$\mathrm{SSE}_{\text{split}} = \mathrm{SSE}(R_{\text{left}}) + \mathrm{SSE}(R_{\text{right}}).$$

We choose the split that *minimizes* $\mathrm{SSE}_{\text{split}}$, or equivalently that maximizes the reduction

$$\Delta\mathrm{SSE} = \mathrm{SSE}(R) - \mathrm{SSE}_{\text{split}}.$$

3.4 Split Criterion: Impurity Decrease

In both classification and regression, we can view a split as choosing a feature j and threshold t that partition node R into

$$R_{\text{left}}(j,t) = \{(x_i, y_i) \in R : x_{ij} \le t\},$$

$$R_{\text{right}}(j,t) = \{(x_i, y_i) \in R : x_{ij} > t\},$$

where x_{ij} is the value of feature j for sample i.

Let $I(R)$ be an impurity measure (Gini, Entropy, or SSE/MSE). The **impurity decrease** from the split is

$$\Delta I(j,t) = I(R) - \left(\frac{N_{\text{left}}}{N_R} I(R_{\text{left}}) + \frac{N_{\text{right}}}{N_R} I(R_{\text{right}}) \right).$$

We search over candidate (j,t) and choose the split that maximizes $\Delta I(j,t)$.

4. Prediction with a Trained Tree

Once a decision tree has been built, making a prediction for a new sample x is straightforward. A key property of decision trees is that prediction does not require evaluating the entire dataset or any optimization; we simply follow the decision rules stored in the tree.

How Prediction Works

To predict the output for a new sample x:

1. Start at the **root node**.
2. At each internal node, check the feature test. For numerical features, this takes the form
$$x_j \leq t,$$
where j is the feature index and t is the threshold chosen during training.
3. Move to the left child if the test is true, otherwise move to the right child.
4. Repeat this process until reaching a **leaf node**.
5. Output the prediction stored in that leaf.

Prediction Values Stored in Leaves

Each leaf stores a simple summary of the training samples that reached it:

- **Classification:** The leaf predicts the majority class:
$$\hat{y}_{\text{leaf}} = \arg\max_k \hat{p}_{kR},$$
where $\hat{p}_{kR}$ is the empirical class frequency in that leaf. Some implementations also store the full class-probability vector $(\hat{p}_{1R}, \dots, \hat{p}_{KR})$, allowing probability estimates.
- **Regression:** The leaf predicts the mean target value:
$$\hat{y}_{\text{leaf}} = \bar{y}_R.$$
This corresponds to the value that minimizes squared error within the region.

Interpretability of Predictions

A major advantage of decision trees is that each prediction comes with a clear explanation. The path followed from the root to a leaf can be read directly as a sequence of human-interpretable rules, such as:

```
If feature_3 ≤ 1.7 → go left, If feature_1 > 5.5 → go right, Predict
class = 2.
```

This makes decision trees one of the most transparent models used in machine learning. Every prediction corresponds to a small set of conditions, making trees valuable for debugging, teaching, and domains where interpretability is essential.

5. Mathematical Algorithm

Algorithm 1 Decision Tree (CART, Top-Down Greedy Splitting)

```
1: Input: training data $\{(x_i, y_i)\}_{i=1}^N$, impurity measure $I(\cdot)$ (Gini/Entropy
   or SSE/MSE), maximum depth max_depth, minimum samples per split
   min_samples_split.
   BuildTree$R$, depth
2: $R :=$ set of indices of samples in this node
3: Compute impurity $I(R)$
4: Stopping condition:
5: if depth $\geq$ max_depth or $|R| <$ min_samples_split or all $y_i$ for $i \in R$ are identical
   then
6:    Create a leaf:
        • classification: predict majority class in $R$
        • regression: predict mean of $y_i$ for $i \in R$
7:    return leaf
8: end if
9: Initialize best impurity decrease $\Delta I^\star \leftarrow 0$
10: Initialize best split $(j^\star, t^\star) \leftarrow$ None
11: for each feature $j$ do
12:    for each candidate threshold $t$ for feature $j$ do
13:       Split $R$ into $R_{\text{left}}$ and $R_{\text{right}}$ using the test $x_{ij} \leq t$
14:       Compute weighted impurity after the split, $I_{\text{split}}$
15:       $\Delta I(j,t) \leftarrow I(R) - I_{\text{split}}$
16:       if $\Delta I(j,t) > \Delta I^\star$ then
17:          $\Delta I^\star \leftarrow \Delta I(j,t)$
18:          $(j^\star, t^\star) \leftarrow (j,t)$
19:       end if
20:    end for
21: end for
22: if $\Delta I^\star = 0$ then
23:    Create and return a leaf (no useful split)
24: end if
25: Create an internal node with test "feature $j^\star \leq$ threshold $t^\star$"
26: Define child index sets $R_{\text{left}}$ and $R_{\text{right}}$ according to this test
27: Recursively build children:
28:    left $\leftarrow$ BuildTree$(R_{\text{left}}, \text{depth}+1)$
29:    right $\leftarrow$ BuildTree$(R_{\text{right}}, \text{depth}+1)$
30: Attach left and right to this node and return node
31: Output: root node BuildTree$(\{1,\dots,N\}, 0)$
```

6. Decision Tree Algorithms

In this lecture, we use the **CART (Classification and Regression Trees)** algorithm, which is the standard in modern machine learning. CART supports both classification and regression, uses binary splits that fit naturally into ensemble methods, handles continuous and categorical features directly, and is the tree structure adopted by Random Forest, Gradient Boosting, XGBoost, LightGBM, and CatBoost. For these reasons, CART is the only decision tree framework covered in this course.

Other tree-building algorithms exist but are mainly of historical interest: **ID3** (information gain; categorical, multi-way splits), **C4.5** (extends ID3 with continuous features and gain ratio), **CHAID** (chi-square–based multi-way categorical splits), and **QUEST** (uses statistical tests to reduce selection bias). While influential, these methods are not used in modern ML workflows, where CART remains the dominant and most practical approach.

7. Summary

Decision Trees:

- Recursively partition the input space using simple threshold tests on features.
- Use impurity measures (Gini, Entropy, or SSE/MSE) to choose splits that make child nodes more homogeneous than the parent node.
- Provide transparent and interpretable models where each path from root to leaf corresponds to a human-readable rule.
- Serve as fundamental building blocks for more advanced ensemble methods such as Random Forest and Gradient Boosting.

Although single trees can overfit and may be unstable under small data perturbations, their simplicity and interpretability make them an essential concept in machine learning.