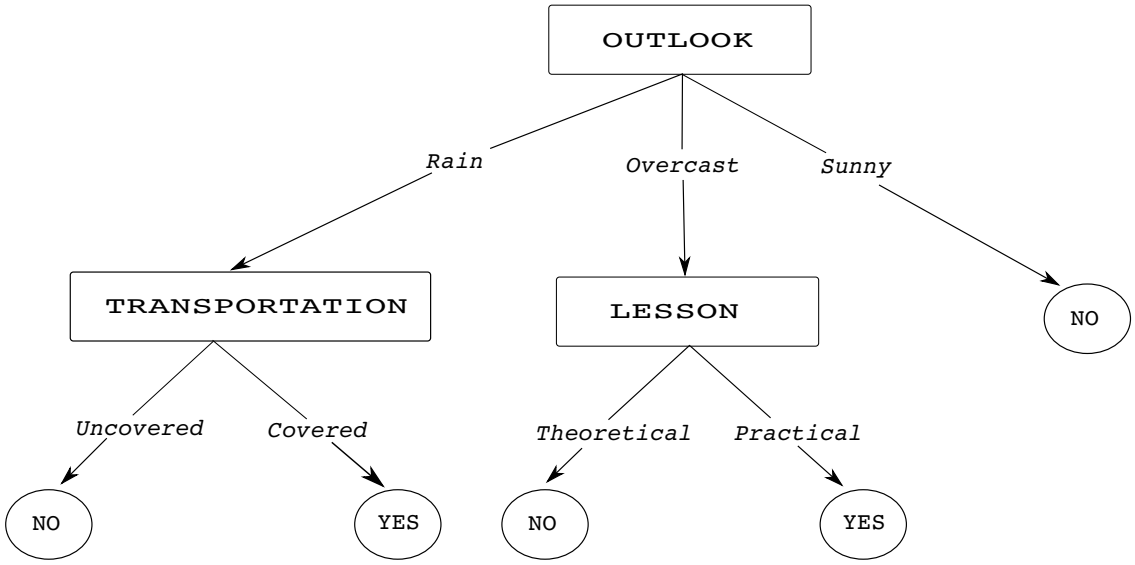
Learning the concept *Go to lesson*



Decision trees encode logical formulas

- A decision tree represents a disjunction of conjunctions of constraints over attribute values.
- Each path from the root to a leaf is a conjunction of the constraints specified along it, for example

$$\text{OUTLOOK} = \text{Overcast} \wedge \text{LESSON} = \text{Theoretical}.$$

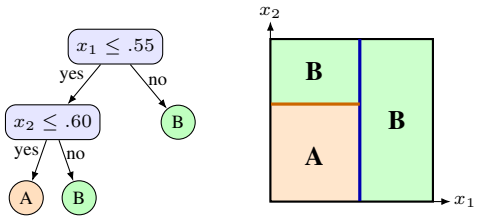
- The leaf contains the prediction assigned to instances reaching it.
- The complete tree is the disjunction of the rules associated with all root-to-leaf paths.

Interpretability
A prediction can be explained by reporting the sequence of tests followed from the root to the corresponding leaf.

Why decision trees?
Decision trees are attractive because they

- model nonlinear decision boundaries;
- require little feature preprocessing;
- naturally express interactions between attributes;

- produce predictions that can be explained as rules;
- support classification and regression.



Key idea
A decision tree recursively partitions the input space into regions that can be assigned simple predictions.

Trees partition the feature space

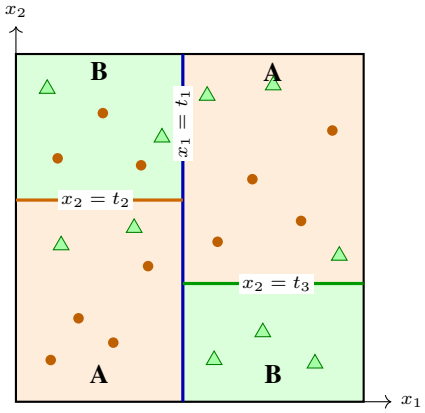
Each internal node introduces a test such as

$$x_j \leq t.$$

The test divides one region into two smaller regions.

A root-to-leaf path therefore identifies one region of the input space.

- Classification tree: predict a class in each region.
- Regression tree: predict a numerical value in each region.



Consequence
Standard decision trees create axis-aligned, piecewise-constant predictors.

Appropriate problems for decision trees

- Binary or multiclass classification; extensions to regression exist.
- Instances represented by numerical or categorical attributes.
- Concepts that can be described by alternative rules or interactions.
- Problems where little preprocessing is desirable.

- Applications requiring an interpretable prediction mechanism.
- Data with missing values, provided an explicit handling strategy is used.

Preprocessing
Unlike distance-based and linear methods, trees do not require feature standardization and are invariant to monotone transformations of individual numerical attributes.

Learning decision trees

- Greedy top-down strategy: ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and CART (Breiman et al., 1984).
- At each node, starting from the root with the full training set:
  1. choose the split producing the best improvement;
  2. create the corresponding child nodes;
  3. partition the node training examples among the children;
  4. recursively repeat the procedure;
  5. stop according to a purity, size, depth, or improvement criterion.
- This is a *divide et impera* approach.

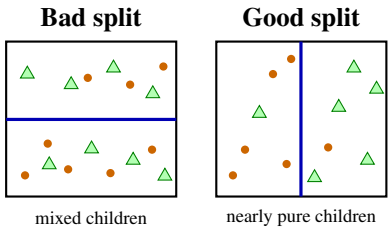
Greedy does not mean globally optimal
The best split at the current node need not belong to the globally best tree. Finding a globally optimal tree is generally computationally difficult.

What makes a good split?
Before the split:

- examples with different labels are mixed;
- the class of a new example is uncertain.

After a good split:

- child nodes are more homogeneous;
- class uncertainty is reduced;
- simple leaf predictions become possible.



Split criterion
Quantify node impurity before and after the candidate split.

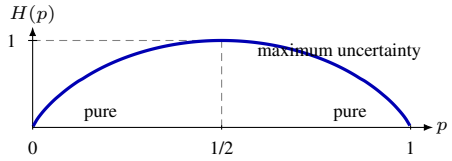
Choosing the best attribute

Entropy
For a collection of labelled examples S with c possible classes,

$$H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i,$$

where p_i is the fraction of examples in class i.

- $H(S) = 0$ when all examples belong to the same class.
- Entropy is maximal when the class distribution is uniform.
- Entropy measures label uncertainty, or equivalently node impurity.



Choosing the best attribute

Information gain
The information gain of a split on attribute A is the expected reduction in entropy:

$$IG(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H(S_v),$$

where S_v is the subset of examples taking value v for A.

- The second term is the weighted average impurity after the split.
- A high information gain means that the split produces purer children.
- ID3 chooses the attribute with maximum information gain.

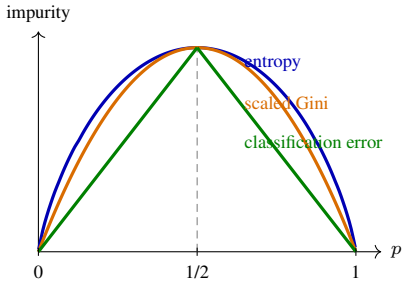
Equivalent view
Choose the split that maximally reduces uncertainty about the label.

Alternative impurity measures
For class proportions $p_1, \ldots, p_c$, common node impurities are

$$\begin{aligned} &\text{Entropy:} && -\sum_{i=1}^{c} p_i \log_2 p_i, \\ &\text{Gini impurity:} && 1 - \sum_{i=1}^{c} p_i^2, \\ &\text{Classification error:} && 1 - \max_i p_i. \end{aligned}$$

- Entropy and Gini are sensitive to changes in class proportions.
- Classification error is less sensitive and is rarely preferred for growing the tree.

- CART commonly uses Gini impurity for classification.



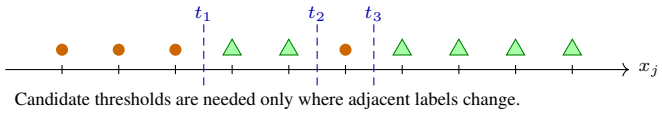
Continuous-valued attributes

A numerical split usually has the form

$$x_j \le t \qquad \text{versus} \qquad x_j > t.$$

Candidate-threshold procedure

1. Sort the examples according to attribute x_j.
2. Consider thresholds between consecutive distinct values, often restricting attention to adjacent examples with different labels.
3. Compute the impurity reduction associated with each threshold.
4. Select the attribute and threshold producing the best reduction.



Candidate thresholds are needed only where adjacent labels change.

Bias toward many-valued attributes

- Information gain can prefer attributes with many possible values.
- An extreme example is a unique identifier: it perfectly separates the training examples but cannot generalize to new data.
- Define the entropy of the partition induced by attribute A:

$$H_A(S) = -\sum_{v \in Values(A)} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|}.$$

- The *gain ratio* penalizes highly fragmented partitions:

$$IGR(S, A) = \frac{IG(S, A)}{H_A(S)}.$$

General lesson

A split that memorizes the training set is not necessarily informative for unseen examples.

Regression trees

Regression trees predict a numerical target rather than a class label.

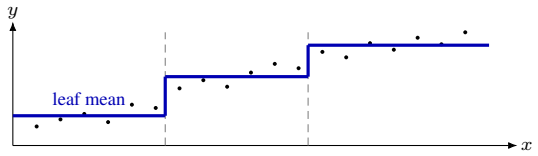
- A leaf commonly predicts the mean target of the training examples reaching that leaf:

$$\hat{y}_R = \frac{1}{|S_R|} \sum_{i \in S_R} y_i.$$

- A candidate split is evaluated by its reduction in squared error:

$$\sum_{i \in S} (y_i - \bar{y}_S)^2 - \sum_{R \in \{L,R\}} \sum_{i \in S_R} (y_i - \bar{y}_R)^2.$$

- The resulting predictor is piecewise constant.



Overfitting in decision trees

- Requiring every leaf to be pure can produce a very large tree.
- A sufficiently deep tree can memorize noise and isolated examples.
- Large trees usually have low training error but high variance.
- Leaves may therefore remain impure and predict the majority class or the empirical class distribution.

Pre-pruning

Stop growing the tree using criteria such as maximum depth, minimum leaf size, or minimum impurity decrease.

Post-pruning

Grow a large tree and remove subtrees whose additional complexity is not supported by validation performance or a complexity penalty.

Reduced-error pruning

- Post-pruning strategy requiring a separate labelled validation set.

Procedure

1. For each internal node, evaluate validation performance after replacing its subtree with a leaf.
2. If every candidate removal worsens validation performance, stop.
3. Otherwise prune the subtree giving the largest improvement, or the smallest degradation allowed by the chosen rule.
4. Assign the replacement leaf the majority class of the corresponding training examples.
5. Repeat.

Trade-off

Pruning sacrifices some training fit to reduce variance and improve generalization.

Handling missing values

Suppose example x has a missing value for the attribute tested at node n.

Imputation Replace the missing value using a statistic estimated from the training data.

Explicit missing branch Treat "missing" as an additional value.

Fractional propagation Send the example to several children with weights proportional to the observed branch frequencies.

Avoid leakage

Any imputation rule must be estimated using training data only and then applied unchanged to validation and test examples.

Strengths and limitations of a single tree

Strengths

- Interpretable rules
- Nonlinear interactions
- Mixed feature types
- Little preprocessing
- Fast prediction
- Embedded feature selection

Limitations

- Greedy optimization
- High variance and instability
- Axis-aligned splits
- Piecewise-constant predictions
- Bias toward dominant or many-valued features
- Poor extrapolation in regression

Instability

Small changes in the training data can produce a substantially different tree, motivating ensemble methods.

Random forests

Training

1. Given N training examples, draw a bootstrap sample of size N.
2. Train a decision tree on the sample; at each node, consider only a random subset of m features when choosing the split.
3. Repeat to obtain a forest of M trees.

Prediction

- Classification: return the majority vote, or average class probabilities.
- Regression: average the numerical predictions.

Why it works
Averaging many accurate but imperfectly correlated trees reduces variance.

From decision trees to ensembles

Decision tree
$\Downarrow$
Bagging
$\Downarrow$
Random forest
$\Downarrow$
Gradient boosting
$\Downarrow$
XGBoost, LightGBM, CatBoost

Modern perspective
A single tree is often used for interpretability. Ensembles of trees are among the strongest general-purpose methods for structured, tabular data.

Different strategies
Bagging and random forests mainly reduce variance; boosting builds trees sequentially to correct previous errors and reduce bias.

Take-home messages

1. Decision trees recursively partition the input space into simple prediction regions.
2. Split criteria such as information gain and Gini impurity reward purer child nodes.
3. Tree induction is greedy and can overfit; stopping and pruning control model complexity.
4. Classification and regression trees use the same recursive-partition principle with different impurity or error measures.
5. Single trees are interpretable but unstable.
6. Random forests and boosting turn trees into highly competitive models for tabular data.