STAT 24620=FINM 34700, STAT 32950
Multivariate Data Analysis
Lecture 12: Decision Trees

Jingshu Wang

The University of Chicago

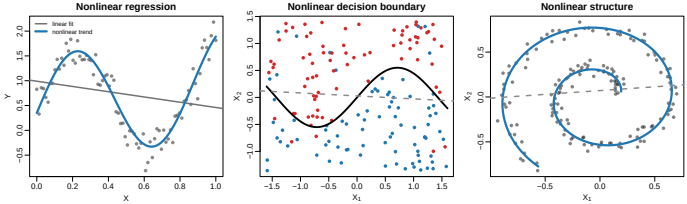
Outline

From linear to nonlinear methods

- Many methods we have used are linear in an important sense:

$$\hat{f}(x) = \beta_0 + x^\top \beta, \quad z = v^\top x.$$

- Linear structure is stable and interpretable.
- But it can miss important nonlinear structure:
  - nonlinear regression functions;
  - nonlinear decision boundaries;
  - curved clusters or manifolds.

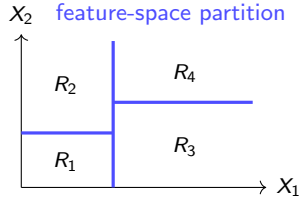


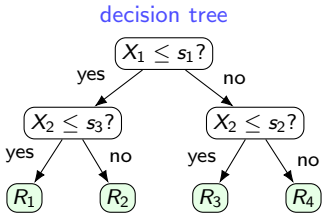
Today: decision trees as a supervised nonlinear method built from simple, interpretable pieces.

The basic idea: recursive splitting

- Start with all observations in one region.
- Split one region into two child regions.
- Repeat inside one child region at a time.
- Each terminal region becomes a leaf.

This is the generic tree structure; the leaf prediction depends on whether the response is quantitative or categorical.





Two types of decision trees

Regression tree

- Response Y is quantitative.
- Each leaf predicts a number:

$$\hat{f}(x) = \bar{y}_m, \qquad x \in R_m.$$

- Splits are chosen to reduce within-node squared error.

Classification tree

- Response Y is categorical.
- Each leaf predicts class probabilities:

$$\hat{p}_{mk} = \Pr(Y = k \mid x \in R_m).$$

- Splits are chosen to reduce class impurity.

Same recursive partition; different leaf summaries and split criteria.

CART: classification and regression tree

Why trees are attractive

- They automatically capture interactions.
- They handle nonlinear relationships without specifying basis functions.
- The fitted rule can be displayed as a sequence of decisions.
- They work for both regression and classification.

But: a single tree is often unstable. Small changes in the data can produce different early splits.

Today's goal: understand how a tree is built and why it needs regularization.

A regression tree model

Suppose we partition the predictor space into regions $R_1, \ldots, R_M$.

A regression tree predicts

$$\hat{f}(x) = \sum_{m=1}^{M} c_m \mathbf{1}\{x \in R_m\},$$

where

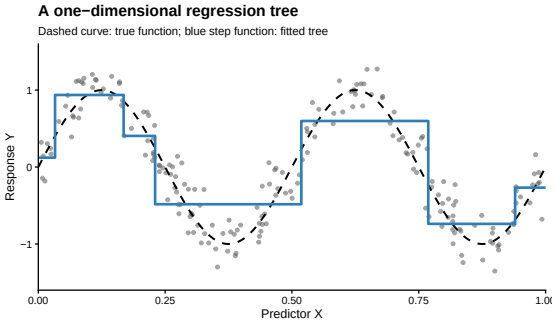
$$c_m = \frac{1}{|\{i : x_i \in R_m\}|} \sum_{i:x_i \in R_m} y_i.$$

- The function is piecewise constant.
- Complexity is controlled by the number and depth of the regions.
- Fitting the best possible partition is computationally hard, so CART uses greedy recursive splitting.

A regression tree in one dimension

- Suppose there is only one predictor X.
- A tree split has the form $X \leq s$ versus $X > s$.
- Repeated splits divide the line into intervals $R_1, \ldots, R_L$.
- Each leaf predicts the average response in its interval:

$$\hat{f}(x) = \sum_{\ell=1}^{L} \bar{Y}_\ell \, \mathbf{1}\{x \in R_\ell\}.$$



In one dimension, a decision tree is a data-adaptive step function.

How does a regression tree choose a split?

At a node containing observations N, consider a split

$$X_j \le s \quad \text{vs.} \quad X_j > s.$$

The CART regression split minimizes the within-node sum of squares:

$$\sum_{i \in N_L(j,s)} (y_i - \bar{y}_L)^2 + \sum_{i \in N_R(j,s)} (y_i - \bar{y}_R)^2.$$

- Try many variables j and cutpoints s.
- Choose the split with the largest reduction in squared error.
- Then repeat recursively inside each child node.

This is greedy: the best current split may not lead to the best final tree.

One split as variance reduction

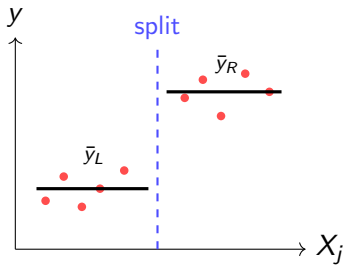
For a node N, define

$$RSS(N) = \sum_{i \in N} (y_i - \bar{y}_N)^2.$$

The split improvement is

$$\Delta(j, s) = RSS(N) - RSS(N_L) - RSS(N_R).$$

- Good splits create child nodes with more homogeneous responses.
- This is analogous to reducing within-cluster variation.
- But the tree is supervised: the response y determines the split.



What can a split look like?

Quantitative predictor

$$X_j \leq s \quad \text{vs.} \quad X_j > s.$$

- Try candidate cutpoints s.
- Choose the cutpoint with the largest improvement.
- The same predictor can be split again later, possibly with a different cutpoint.

Categorical predictor

$$X_j \in A \quad \text{vs.} \quad X_j \notin A,$$

where A is a subset of categories.

- Example: {red, blue} vs. {green}.
- CART still makes a binary split.

Each split uses one variable, but variables are not "used up" after a split.

Classification trees

For classification, each terminal region estimates class probabilities:

$$\hat{p}_{mk} = \frac{1}{n_m} \sum_{i: x_i \in R_m} \mathbf{1}\{y_i = k\}.$$

The predicted class is often

$$\hat{C}(x) = \arg\max_k \hat{p}_{mk}, \qquad x \in R_m.$$

- Leaves can output class labels or probabilities.
- Probabilities are useful for ROC curves, thresholds, and uncertainty.
- Splits are chosen to make child nodes more class-homogeneous.

Node impurity

Let $\hat{p}_{mk}$ be the class proportion for class k in node m.

Gini impurity

$$G_m = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2.$$

Entropy

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk}).$$

- Both are small when a node is mostly one class.
- CART commonly uses Gini impurity.
- A good split produces child nodes with lower weighted impurity.

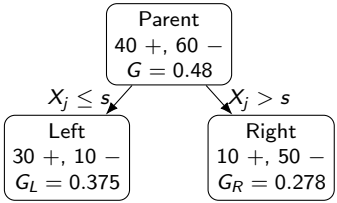
A binary-node example

Before splitting

$G_{\text{parent}} = 1 - 0.4^2 - 0.6^2 = 0.48.$

Candidate split

node	$+/-$	n_m	G_m
L	30/10	40	0.375
R	10/50	60	0.278



After splitting

$$G_{\text{after}} = \frac{40}{100} G_L + \frac{60}{100} G_R$$
$$= 0.4(0.375) + 0.6(0.278) = 0.317.$$

$\Delta = G_{\text{parent}} - G_{\text{after}} = 0.48 - 0.317 = 0.163.$

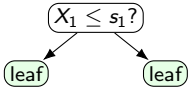
Since $\Delta > 0$, this split reduces impurity; we prefer splits with larger Δ.

Tree depth controls flexibility

- Depth = longest path from the root to a leaf.
- A depth-d binary tree has at most 2^d terminal regions.
- Shallow trees use large regions and average over heterogeneous observations.
  - A coarse approximation to $f(x)$, hence larger bias.
- Deep trees use small leaves; predictions may depend on few observations.
  - More local adaptation, but larger variance and instability.

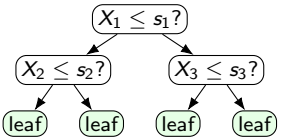
Depth controls the bias–variance tradeoff by controlling how small the terminal regions can become.

shallow tree



few large regions
stable, less adaptive

deeper tree

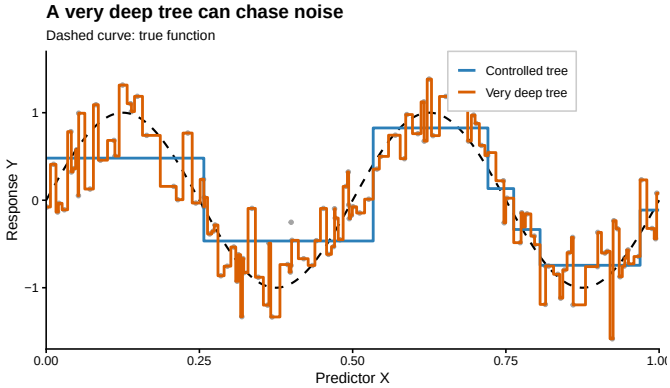


many small regions
adaptive, less stable

Why not grow the best possible large tree?

- A very deep tree keeps splitting until leaves are small.
- In regression, a tiny leaf predicts an average from only a few observations.
- Training RSS can be very small because the tree follows random noise.
- But the fitted function becomes jagged and sample-sensitive.

Low training error is not the goal; we want stable prediction on new data.



Cost-complexity pruning: the objective

Cost-complexity pruning chooses the tree that minimizes

$$C_\alpha(T) = L(T) + \alpha|T|,$$

where $|T|$ is the number of terminal nodes (leaves).

Regression tree

$$L(T) = \sum_{m=1}^{|T|} \sum_{i:x_i \in R_m} (y_i - \bar{y}_m)^2.$$

Classification tree

$$L(T) = \sum_{m=1}^{|T|} n_m \, Q_m,$$

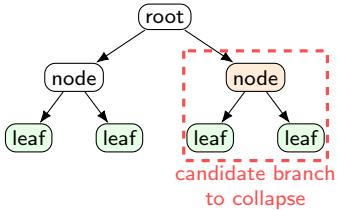
where Q_m measures impurity or prediction error in leaf m.

- Larger α favors smaller trees.
- α is typically chosen by cross-validation.
- For classification, splits may use Gini while pruning/CV uses classification error.

This is regularization: accept a little bias to reduce variance.

Weakest-link pruning: intuition

- First grow a large tree T_0.
- Consider collapsing each internal node into a leaf.
- Collapsing removes all descendants below that node.
- Each collapse trades off:
  - increase in loss
  - reduction in number of leaves
- We collapse the branch with the smallest loss increase per leaf removed.



This is the weakest-link: the easiest branch to prune.

Weakest-link pruning: connecting to the objective

Recall the cost-complexity objective:

$$C_\alpha(T) = L(T) + \alpha|T|$$

Consider pruning a subtree T_t into a single leaf t.

- $L(T_t)$ is the loss of the subtree, and $L(t)$ is the loss if we collapse it into one leaf.
- Before pruning:

$$C_\alpha(T) = L(\text{rest}) + L(T_t) + \alpha(|T_t| + \text{number of rest leaves})$$

- After pruning:

$$C_\alpha(T') = L(\text{rest}) + L(t) + \alpha(1 + \text{number of rest leaves})$$

Subtract:

$$C_\alpha(T') - C_\alpha(T) = \big(L(t) - L(T_t)\big) - \alpha(|T_t| - 1)$$

Prune if this is negative

Weakest-link pruning: which branch is weakest?

From the previous slide, pruning is beneficial if:

$$L(t) - L(T_t) \leq \alpha(|T_t| - 1)$$

Rearrange:

$$\alpha \geq \frac{L(t) - L(T_t)}{|T_t| - 1} \equiv \alpha_t$$

- α_t is the smallest penalty at which pruning this branch is worthwhile.
- We prune the branch with the smallest α_t first.

Repeating this produces a nested sequence of trees

$$T_0 \supset T_1 \supset T_2 \supset \cdots$$

- Each T_k is associated with a critical value α_k
- It has been shown that each T_k is optimal for $C_\alpha(T)$ for $\alpha \in [\alpha_k, \alpha_{k+1})$

The tree-building workflow

Step 1: Construct the pruning path (no α chosen)

- Grow a large tree T_0.
- Apply weakest-link pruning to generate a nested sequence of subtrees:

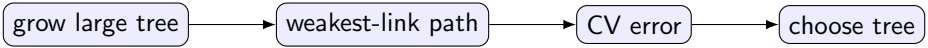
$$T_0 \supset T_1 \supset T_2 \supset \cdots$$

Step 2: Select the best α using CV

- For each fold, compute its weakest-link pruning path on the held-in data.
- For each α, prune the tree and evaluate prediction error on the held-out data.
- Average the prediction error across folds and choose the α with the smallest CV error.

Final step

- Select the subtree T_k from step 1 using the CV selected α.



How to interpret a fitted tree

- The top splits are often the most important for prediction.
- Each leaf describes a subgroup with a fitted mean or class probability.
- A path from root to leaf gives a rule:

$$X_1 \le s_1,\ X_3 > s_3,\ X_2 \le s_2 \quad \Rightarrow \quad \hat{y} = c_m.$$

- Interpretability is strongest for small or moderately sized trees.

A tree is a predictive rule, not automatically a causal explanation.

Strengths and weaknesses

Strengths

- nonlinear and interaction-friendly;
- handles mixed variable types;
- easy to visualize when small;
- little preprocessing needed.

Weaknesses

- unstable to small data changes;
- piecewise-constant predictions;
- often weaker prediction than ensembles;
- greedy fitting can miss better global structure.

A quick R demo

Notebook file: `Lecture13_demo.nb.html`, Tree section.

Connection to previous lectures

- Like clustering, trees partition observations into groups.
- Unlike clustering, tree partitions are chosen to predict a response.
- Like regularized regression, trees need complexity control.
- Unlike linear models, trees can represent interactions without explicit interaction terms.

Trees give a flexible nonlinear base learner; next time, we stabilize them by averaging many trees.

Lecture 12 summary

- A decision tree partitions feature space by recursive binary splits.
- Regression trees minimize within-node squared error; classification trees reduce impurity.
- Tree depth and pruning control the bias–variance tradeoff.
- Single trees are interpretable but often unstable.
- This instability motivates ensemble methods such as bagging and random forests.

Suggested reading

- James, Witten, Hastie, Tibshirani (2nd edition), Chapter 8.1, 8.3.1, 8.3.2.