Tree-Based Methods

Decision Trees, Bagging, Random Forests, and Boosting

The Big Picture

Tree-based methods work by dividing the predictor space into simple regions and making predictions based on the training observations in each region. While individual trees are easy to interpret, they often lack predictive accuracy. The solution? **Ensemble methods** - combining many trees to dramatically improve performance.

Key insight: A single tree is like asking one expert. Ensemble methods are like consulting a panel of experts and aggregating their opinions.

1. The Basics of Decision Trees

1.1 Regression Trees

Goal: Predict a quantitative (continuous) response.

How It Works

The algorithm divides the predictor space into J distinct, non-overlapping regions (boxes) R1, R2, ..., RJ. For every observation falling into region Rj, the prediction is simply the **mean response value** of the training observations in that region.

Construction: Recursive Binary Splitting

Finding the optimal partition is computationally infeasible, so we use a **top-down, greedy** approach:

* **Top-down**: Start with all observations in one region, then successively split
* **Greedy**: At each step, make the best split for that step without looking ahead

The Splitting Rule: At each step, select predictor Xj and cutpoint s that minimizes the RSS:

$$\min_{j,s}\left[\sum_{i:x_i\in R_1(j,s)}(y_i-\hat{y}_{R_1})^2+\sum_{i:x_i\in R_2(j,s)}(y_i-\hat{y}_{R_2})^2\right]$$

This process repeats within each resulting region until a stopping criterion is met (e.g., no region contains more than 5 observations).

Tree Pruning: Avoiding Overfitting

Large trees overfit (low bias, high variance). Smaller trees have higher bias but lower variance. We need a principled way to find the right size.

Cost Complexity Pruning (Weakest Link Pruning):

Rather than checking every possible subtree, we find a sequence of trees indexed by a tuning parameter alpha. For each alpha, we find the subtree T that minimizes:

$$\sum_{m=1}^{|T|}\sum_{i\in R_m}(y_i-\hat{y}_{R_m})^2+\alpha|T|$$

* |T| = number of terminal nodes (leaves)
* alpha controls the trade-off between fit (first term) and complexity (second term)
* alpha = 0 gives the full tree; large alpha gives a small tree

Selection: Use K-fold cross-validation to choose optimal alpha, then select the corresponding subtree.

1.2 Classification Trees

Goal: Predict a qualitative (categorical) response.

Prediction: Each observation is assigned to the **most commonly occurring class** among training observations in its region.

Splitting Criteria

RSS doesn't make sense for classification. Instead, we measure **node purity** - how homogeneous each node is. Let p_mk be the proportion of observations in region m belonging to class k.

1. Classification Error Rate

$$E = 1 - \max_k(\hat{p}_{mk})$$

Fraction of observations not in the most common class. Simple but not sensitive enough for tree growing.

2. Gini Index

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

Measures total variance across K classes. Small value = node is 'pure' (mostly one class).

3. Entropy (Cross-Entropy)

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

Alternative to Gini. Also small when node is pure.

Practice: Use Gini or Entropy to grow the tree (more sensitive to purity). Use classification error rate for pruning the final tree.

1.3 Advantages and Disadvantages

Advantages	Disadvantages
Very easy to explain and interpret	Lower predictive accuracy than best methods
Mirrors human decision-making	Non-robust: small data changes cause large tree changes
Can be displayed graphically	High variance
Handles qualitative predictors naturally	Greedy splits may miss globally optimal trees

2. Ensemble Methods

The key insight: Individual trees have high variance. By combining many trees, we can dramatically reduce variance while maintaining (or improving) accuracy.

2.1 Bagging (Bootstrap Aggregation)

The Problem: Decision trees have high variance - small changes in data lead to very different trees.

The Solution: Averaging reduces variance. But we only have one training set...

The Trick: Use the **bootstrap** to simulate having multiple training sets!

The Algorithm:

1. Generate B bootstrapped training sets (sample n observations with replacement)
2. Train a deep, unpruned tree on each bootstrapped set
3. Aggregate predictions:

For regression: Average the B predictions

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

For classification: Take a **majority vote** among the B trees

Out-of-Bag (OOB) Error Estimation

On average, each bootstrap sample uses only about 2/3 of the observations. The remaining 1/3 are 'out-of-bag' (OOB).

For each observation i, predict using only the trees where i was OOB. This gives a valid estimate of test error **without cross-validation**!

Variable Importance

Bagging improves accuracy but destroys interpretability. To recover some insight, we compute **variable importance**: the total decrease in RSS (or Gini) due to splits on each predictor, averaged over all B trees.

2.2 Random Forests

The Problem with Bagging:

If one predictor is very strong, all bagged trees will use it for the first split. The trees become **correlated**, and averaging correlated quantities doesn't reduce variance as effectively.

The Solution: Force trees to be different by **decorrelating** them.

The Random Forest Twist:

At each split, randomly select only **m predictors** (out of p total) as candidates. The tree can only split on one of these m.

Typical Choice of m	Rationale
m = sqrt(p) for classification	Good default that works well in practice
m = p/3 for regression	Slightly more predictors for regression
m = p (all predictors)	Reduces to standard bagging

Why it works: By forcing trees to consider different predictors, we ensure they make different splits. Different trees capture different aspects of the data. When averaged, the noise cancels out and the signal remains.

2.3 Boosting

A Different Philosophy:

Bagging/Random Forests: Build trees **independently** on bootstrapped data, then average.

Boosting: Build trees **sequentially** on the original data, where each tree learns from the mistakes of its predecessors.

The Boosting Algorithm (for regression):

```
Set f_hat(x) = 0 and residuals r_i = y_i for all i
For b = 1, 2, ..., B:
(a) Fit a tree with d splits to the training data (X, r) - note: fitting to residuals!
(b) Update the model: add a shrunken version of this tree
(c) Update residuals: r_i = y_i - f_hat(x_i)
Output the final boosted model
```

The update step:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

where lambda is the **shrinkage parameter** (learning rate), typically 0.01 or 0.001.

Boosting Tuning Parameters

Parameter	Description	Guidance
B (# trees)	Number of boosting iterations	Can overfit if too large. Use CV.
lambda (shrinkage)	Learning rate	Small values (0.01, 0.001) work best but need more trees
d (interaction depth)	Number of splits per tree	d=1 (stumps) often works. Larger d for interactions.

Key difference from Bagging: Boosting can overfit if B is too large. Bagging does not overfit as B increases (more trees only helps). Always use cross-validation for boosting!

2.4 BART (Bayesian Additive Regression Trees)

BART takes a Bayesian approach using MCMC (Markov Chain Monte Carlo) to explore the space of possible tree ensembles.

How it works:

1. Start with an initial ensemble of trees
2. In each MCMC iteration, randomly **perturb** trees (add/prune branches, change predictions)
3. Accept changes that improve fit to the partial residuals
4. Discard initial 'burn-in' samples; average over remaining samples

Strength: Often has impressive 'out-of-the-box' performance with minimal tuning.

3. Summary: Comparing Methods

Method	Data Sampling	Tree Growth	Primary Goal
Single Tree	Original data	One pruned tree	Interpretability
Bagging	Bootstrap samples	Independent, deep trees	Reduce variance
Random Forests	Bootstrap + random predictors	Independent, decorrelated	Further reduce variance
Boosting	Original data	Sequential, fit residuals	Reduce bias and variance
BART	Original data	Sequential perturbations	Explore model space (Bayesian)

Intuitive Analogy: Predicting the Weather

Decision Tree: You ask one weather expert. They're smart but prone to errors based on quirks in their training.

Bagging: You ask 100 experts who studied the same books but memorized different chapters (bootstrapped data). You take the average of their predictions.

Random Forests: You ask 100 experts, but force each to look at only a random subset of weather indicators. One looks only at wind, another only at clouds. This prevents them from all copying the 'obvious' forecast.

Boosting: You ask one expert. Then ask a second to fix the first's mistakes. Then a third to fix the second's mistakes. Each new expert focuses specifically on what previous experts got wrong.

4. Key Takeaways

* **Single trees** are interpretable but have high variance and lower accuracy.
* **Bagging** reduces variance by averaging many trees trained on bootstrapped samples.
* **Random Forests** improve on bagging by decorrelating trees (random predictor subsets at each split).
* **Boosting** builds trees sequentially, each learning from predecessors' errors. Can reduce both bias and variance.
* **OOB error** provides free cross-validation for bagging and random forests.
* **Variable importance** helps recover interpretability in ensemble methods.
* Boosting requires careful tuning (can overfit); Random Forests are more robust to hyperparameters.
* For pure prediction accuracy: Random Forests or Boosting usually win.
* For interpretability: Single trees or examining variable importance.

Quick Reference: Key Formulas

Cost Complexity Pruning (minimize this):

$$\sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

Gini Index (node purity for classification):

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

Bagging Prediction:

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Boosting Update:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{b}(x)$$