Trees: pruning and ensembles

Tom Edinburgh
te269

Slides on the course GitLab/at https://github.com/tedinburgh/ads2023

Today: Pruning and ensembles

- Tree sizes
- Pruning
- Boosting
- Bagging
- Random forests

Questions: halfway through, at the end, or by email (te269)

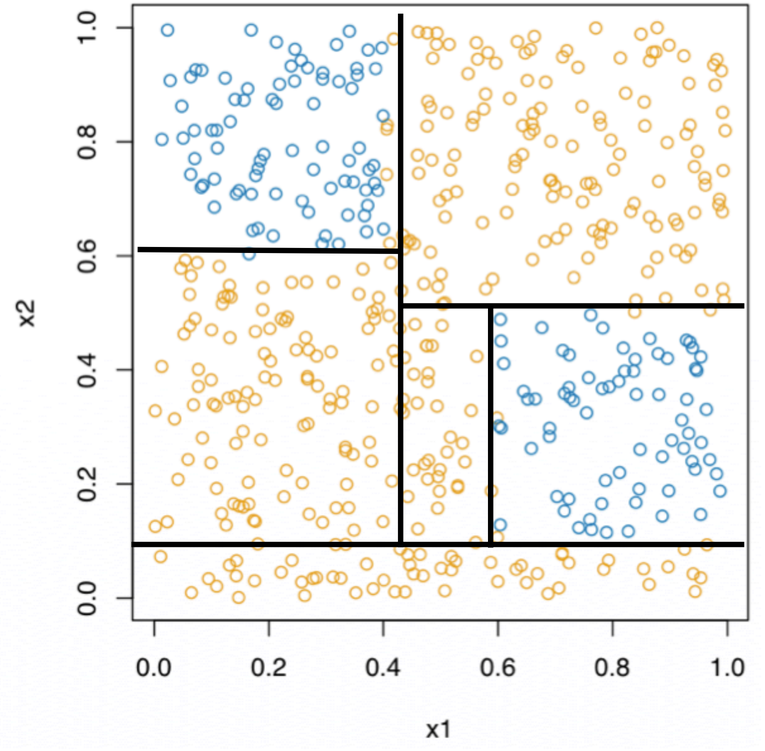
Resources

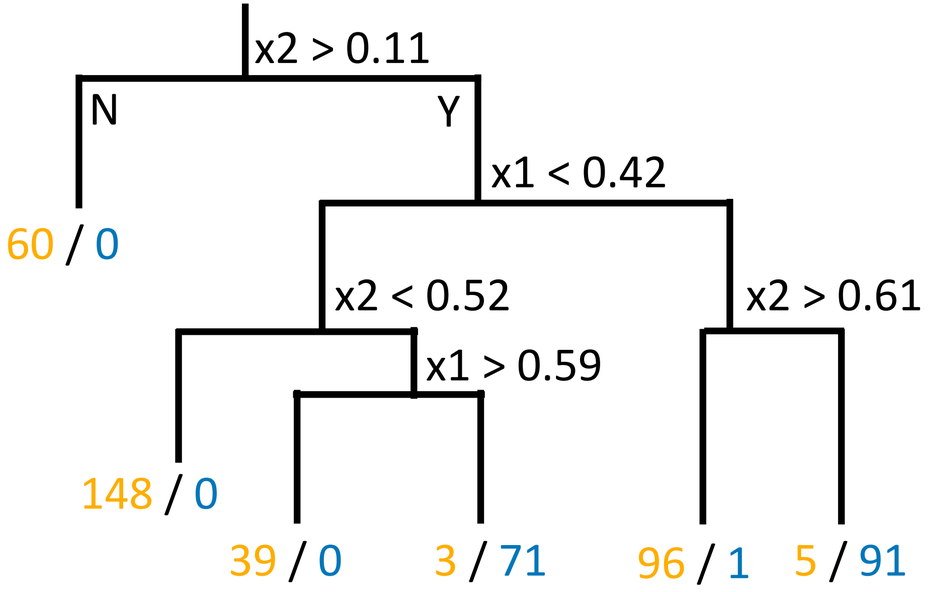
- An Introduction to Statistical Learning with Applications in R/Python (James, Witten, Hastie, Tibshirani, Taylor; 2013/2023)

- Slides adapted from:
  - Prof Alexandra Chouldechova, Carnegie Mellon
  - Prof Stephen Eglen, Cambridge

Recap: Overview

- Tree-based methods **segment** the feature space into simple regions (e.g. high-dimensional rectangles)
- **Predict** using the average (mean, mode), **classify** using the most common class





Recap: How to grow the tree?

- General process:

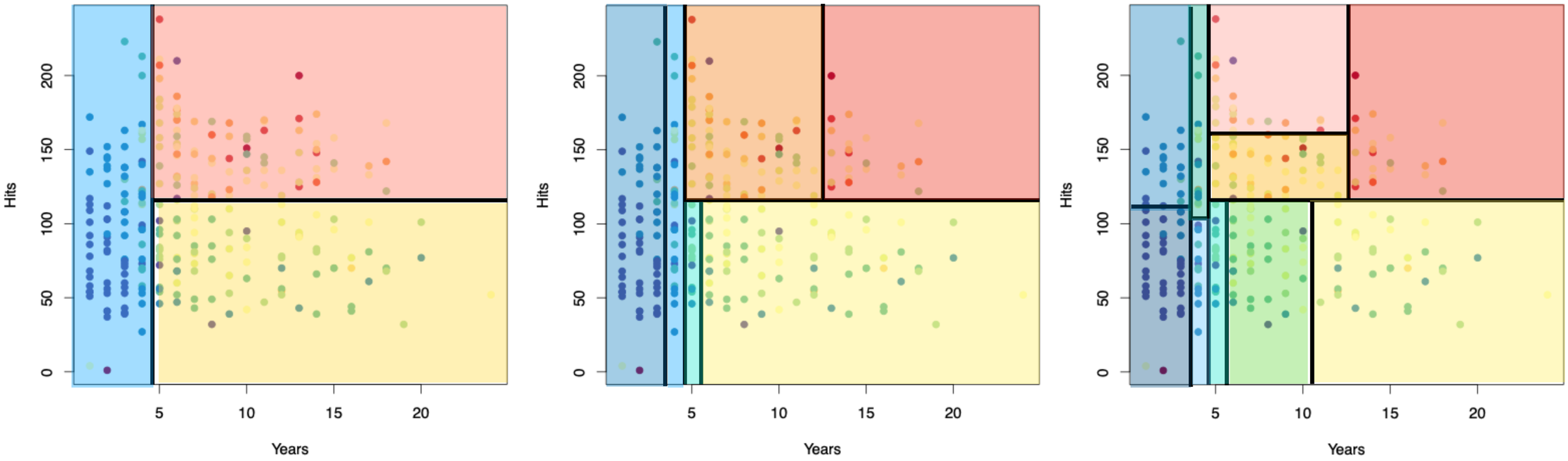
  1. Partition the predictor space (i.e. all possible values of $X_1, X_2, \ldots, X_p$) into J distinct, non-overlapping regions, labelled $R_1, \ldots, R_J$

  2. Each observation in a given region R_j is given the same predicted value (or class), which is the mean (or mode) of all response variables in that region

Recap: When do we stop growing the tree?

- Which is better: small trees or big trees?

Example: baseball salaries



Small trees vs large trees

- Small trees: generalise well but poor prediction, i.e. **underfitting**
- Large trees: good performance on training data but do not generalise well, i.e. **overfitting**
- There's a bias-variance tradeoff between the number of leaves and generalisation to unseen data

	Small	**Large**
Bias	High	Low
Variance	Low	High

Small trees vs large trees

- Small trees: generalise well but poor prediction, i.e. **underfitting**
- Large trees: good performance on training data but do not generalise well, i.e. **overfitting**
- There's a bias-variance tradeoff between the number of leaves and generalisation to unseen data

Bias comes from simplifying assumptions made by the model

Variance is the amount that the model estimate changes if we use slightly different training data

	Small tree	**Large tree**
Bias	High	Low
Variance	Low	High

Small trees vs large trees

- Small trees: generalise well but poor prediction, i.e. **underfitting**
- Large trees: good performance on training data but do not generalise well, i.e. **overfitting**
- There's a bias-variance tradeoff between the number of leaves and generalisation to unseen data

The bias error comes from wrong assumptions (e.g. the problem is more complex than the assumption)

Variance error is error from sensitivity to fluctuations in the training data

	Small tree	**Large tree**
Bias	High	Low
Variance	Low	High

Small trees vs large trees

- We could stop adding new nodes if the decrease in RSS or Gini index is less than a fixed pre-specified threshold
- The problem is that the recursive binary partition algorithm is **greedy**
- We don’t know if a poor partition will be followed immediately a very good one

Pruning

- The solution is to grow a very large tree (lots of leaves), then prune some of the branches

- Start with a large tree T_0 and with complexity hyperparameter α. Find the subtree $T \subset T_0$ that minimises

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

  where $|T|$ is the number of leaves in tree T and R_m is the region associated with the mth leaf

- The two components are the model error term and a penalty on model complexity, similar to penalised regression e.g. Lasso regression

Pruning

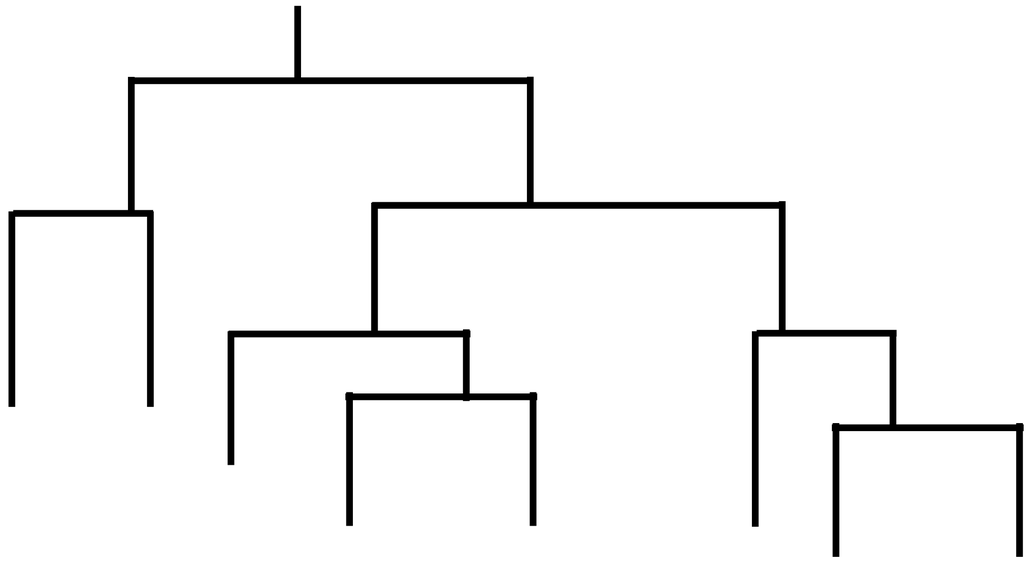
- There may be a lot of subtrees, which will make the computation difficult!
- **Bottom up** vs **top down**

- If we slowly increase α, then the optimal trees are nested (this is useful for computation!)
- If we start with $\alpha = 0$, then the optimal tree is just the full tree T_0
- As $\alpha \to \infty$, the penalty term will mean that the optimal tree only has one leaf
- This is known as **weakest link pruning** or **cost complexity pruning**

Pruning in practice

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

T_0



$\alpha = 0$

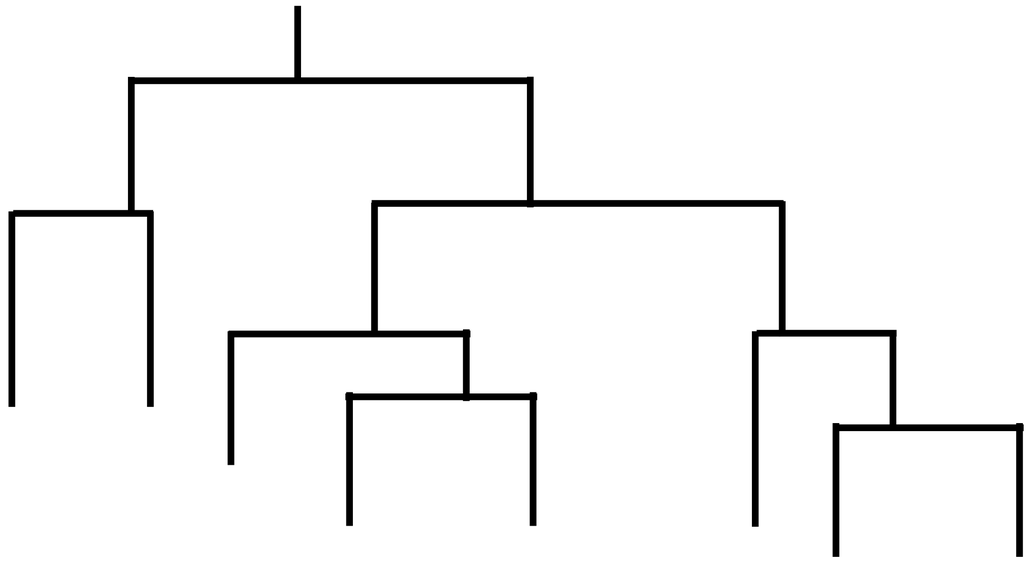
T_∞

$\alpha \to \infty$

Pruning in practice

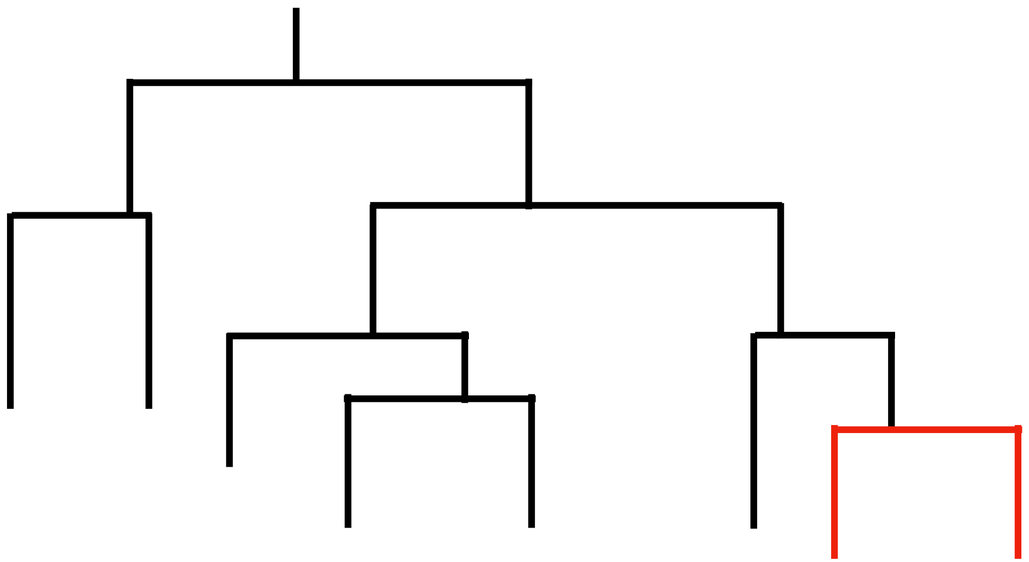
$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

T_0



$\alpha = 0$

T_1

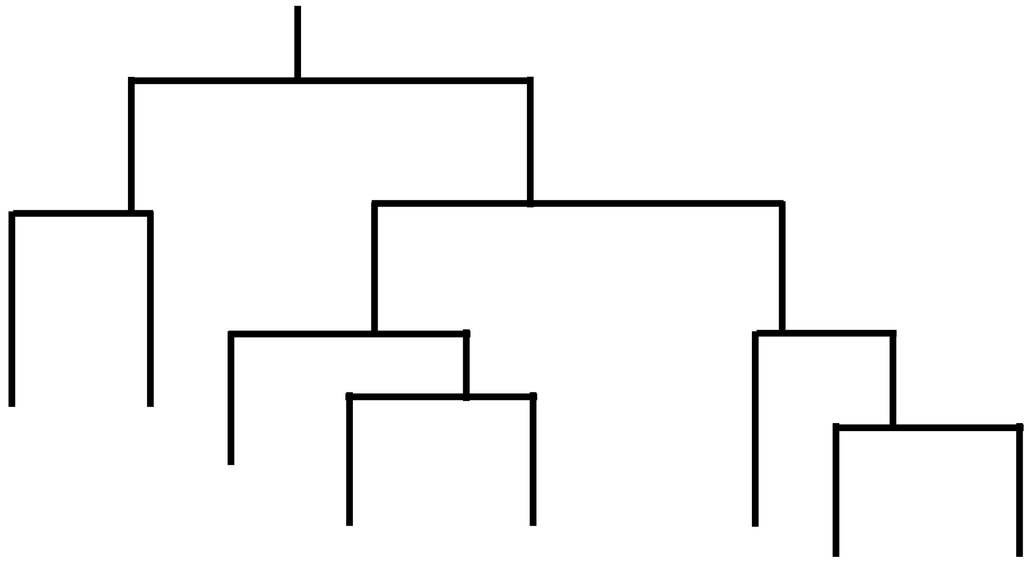


$0 < \alpha < \alpha_1$

Pruning in practice

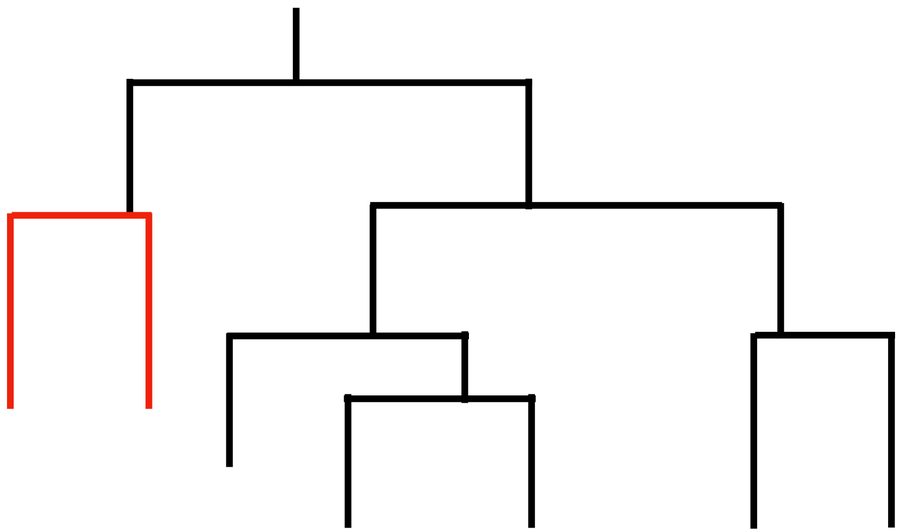
$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

T_0



$\alpha = 0$

T_2

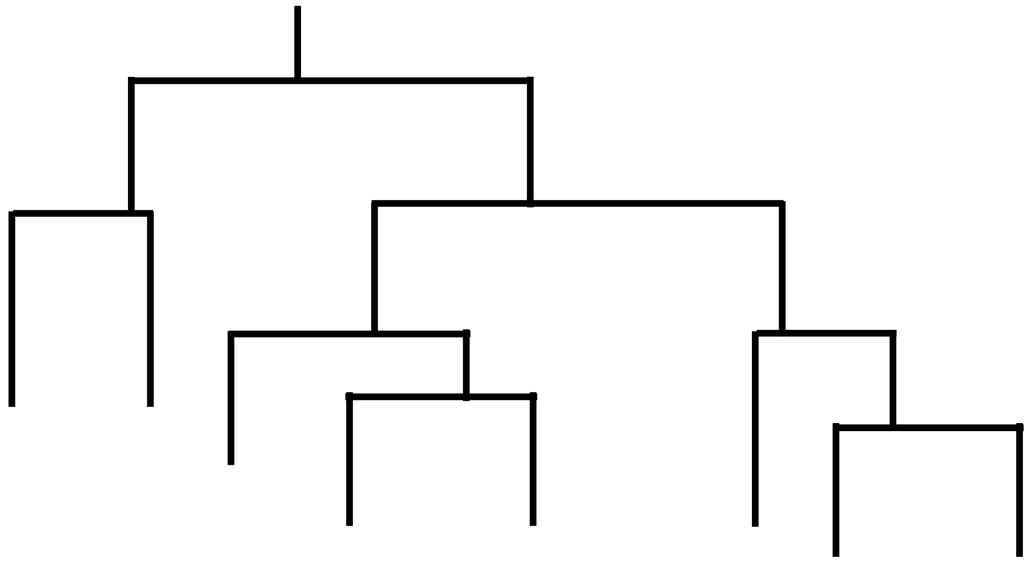


$\alpha_1 < \alpha < \alpha_2$

Pruning in practice

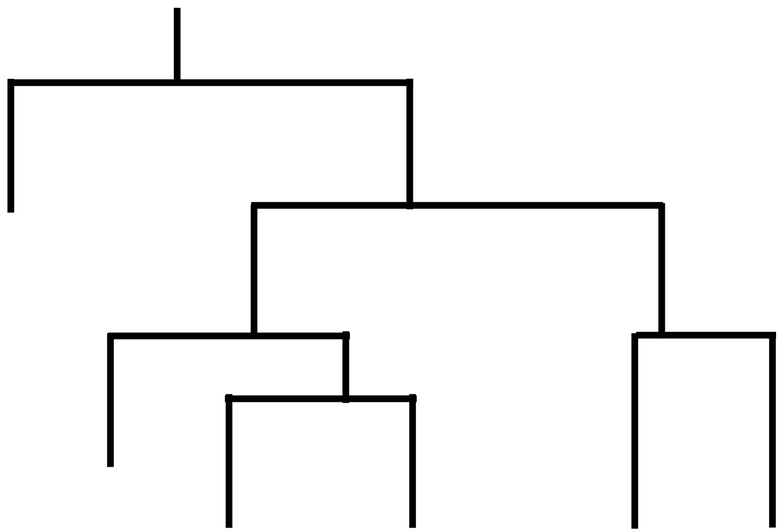
$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

T_0

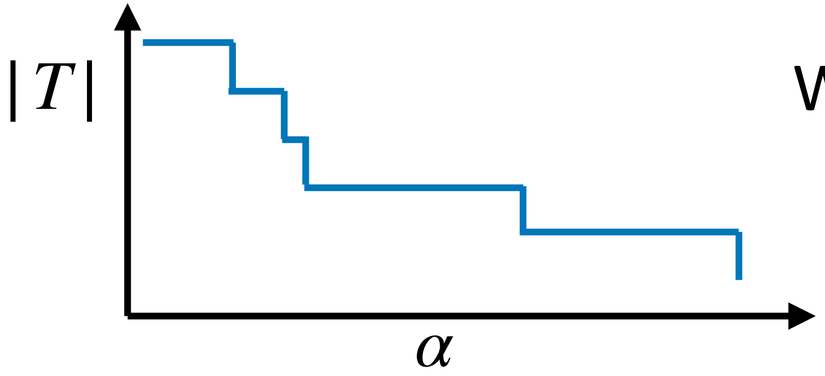


$\alpha = 0$

T_3



$\alpha_2 < \alpha$



What is a good value of α?

Pruning steps

1. Construct a very large tree T_0 using recursive binary partition
2. Apply weakest link pruning to T_0 to create nested sequence of trees as a function of α
3. Use k-fold cross-validation to select the hyper parameter α
   - Split the training data into K folds of equal size
   - For $k = 1, \ldots, K$, repeat steps 1. and 2. on all training data except the kth fold, and evaluate the mean squared prediction error on the kth fold
   - Average the result, then choose α that minimises the average error
4. The optimal tree is the subtree from 2. corresponding to that value of α

Pruning steps

1. Construct a very large tree T_0 using recursive binary partition

2. Apply weakest link pruning to T_0 to create nested sequence of trees as a function of α

3. Use k-fold cross-validation to select the hyper parameter α

   - Split the training data into K folds of equal size

   - For $k = 1, \ldots, K$, repeat steps 1. and 2. on all training data except the kth fold, and evaluate the **mean squared prediction error** on the kth fold

     MSPE is similar to RSS (there are subtle differences between errors and residuals)

   - Average the result, then choose α that minimises the average error

4. The optimal tree is the subtree from 2. corresponding to that value of α

Pruning: code

- Jupyter notebook about regression trees
- (On GitHub https://github.com/tedinburgh/ads2023 and the course GitLab)

Questions?

Decision trees: pros and cons

- Trees are very interpretable! You can easily read off why the model gave a particular prediction. ✓
- Trees handle mixed data types (both predictors and response variables) ✓
- Trees can handle missing data (e.g. using surrogate variables) ✓
- Trees are not very robust. Small changes to the input data can completely change the (unpruned) tree structure. ✘
- Most of the time, trees aren't particularly good at prediction. ✘
- Does this mirror human intuition better than other methods for prediction or classification?

Ensembles

- Predictions from one decision tree on its own may not be very good

- An ensemble method uses many individual decision trees (**weak learners**) as building blocks
- These building blocks can be combined together in a much more powerful model

Bagging and cross-validation

- **Bagging** (bootstrap aggregating) averages across lots of estimates in order to decrease the variance of a high-variance predictor
- Idea: averaging a set of observations reduces the variance (e.g. if $Z_1, \ldots, Z_n$ are independent observations each with variance σ^2, then the variance of the mean $\bar{Z} = 1/n \sum_n Z_n$ is σ^2/n)
- This is similar to cross-validation, which outperforms a single validation set
- A single validation set may give highly variable estimates (pick a different training-validation split and the estimate may change a lot)
- Averaging over multiple folds produces a more stable error

The bootstrap

- This is a very important resampling technique, and is used widely in statistics
- The idea is to use the empirical distribution of the data to estimate the true unknown data-generating distribution

- Sample from the observed data with replacement until you have a dataset of the same size (i.e. sample rows with replacement). This is a bootstrap sample.

$$X = \begin{pmatrix} X_1 & X_2 & \dots & X_p \end{pmatrix} = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

The bootstrap

- Not all of the observations will be in each bootstrap sample
- The probability that row i is in a particular bootstrap sample at least once is $p = 1 - (1 - 1/n)^n \approx 0.632$ (hint: what is the probability that it's not chosen as the first pick?)
- The observations not in the bootstrap sample (called out-of-bag observations) are useful later for validation!

Bagging (bootstrap aggregating)

- Generate B bootstrap samples (sometimes hundreds)
- Build deep trees (no pruning), each on a separate bootstrap sample
- Classification: use **majority voting** (overall prediction is the most common class among all B predictions)
- Regression: take the average, i.e. $\hat{f}_{bag}(x) = \frac{1}{B}\sum_b \hat{f}^{\star b}(x)$

Prediction from decision tree using the bth bootstrap sample

- Out-of-bag (OOB) error is similar to leave-one-out cross-validation error, so can be used model validation.

Probability bagging

- Classification in bagging usually uses **majority voting**

- An alternative is to calculate the bagging estimate of class probabilities

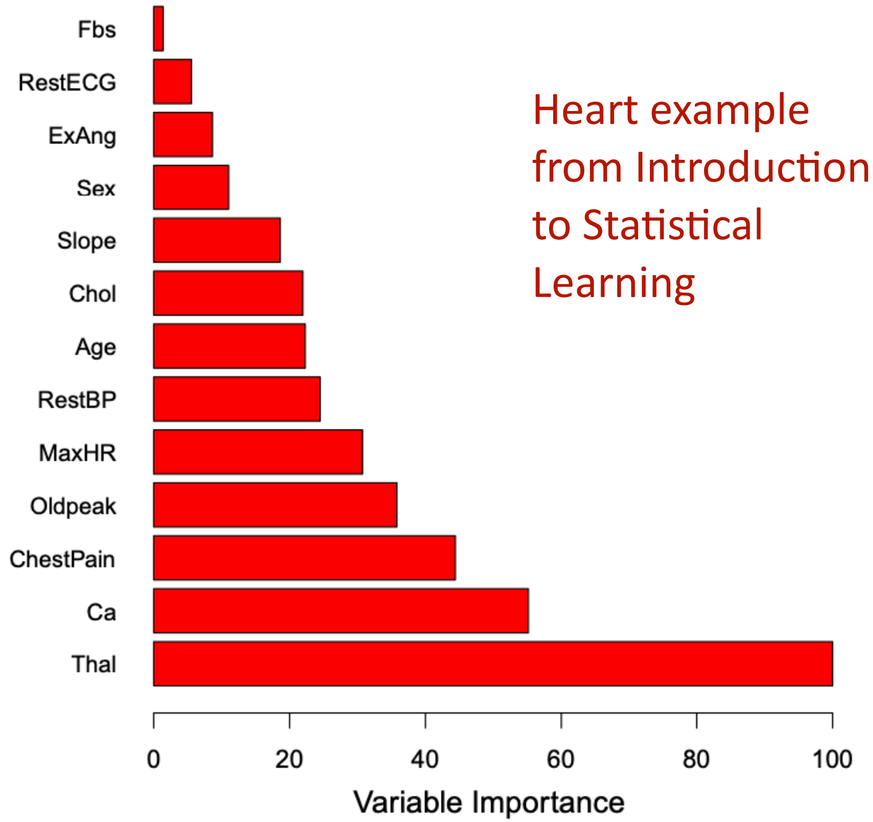
$$\hat{p}_k^{bag}(x) = \frac{1}{B} \sum_b \hat{p}_k^{\star b}(x)$$

- We can classify using $\hat{y}^{bag} = \arg\max_{k=1,\ldots,K} \hat{p}_k^{bag}(x)$

Interpretability

- We lose simple interpretability by aggregating lots of trees
- But we can calculate feature/variable importance as the total amount that the RSS (or Gini index) decreases due to partitions on that variable, averaged over all of the trees

.



Heart example from Introduction to Statistical Learning

Random forests

- Random forests use the same general procedure as bagging, with a small adjustment to decrease correlation between individual trees (this reduces variance when averaging across trees
- This still uses B bootstrap samples (each building its own tree)
- But at every node, we can only partition on a randomly selected subset of m predictors (out of the p predictors) (a rule of thumb is $m \approx \sqrt{p}$)
- Suppose there's one very strong predictor, several reasonably strong predictors, and some not weak predictors - some of the trees will be forced to ignore the very strong predictor in the top split in favour of a reasonably strong predictor

Random forests

- It turns out that increasing the diversity of the trees improves performance!
- Random forests are less likely to get stuck in local optima and explore more of the model space

Boosting

- Boosting also involves combining trees, but there's no bootstrapping involved
- The idea is learn a sequence of **weak learners** (individual trees) that learn slowly
- Each small tree (sometimes just a stump) learns the residuals from the trees before it

- Use cross-validation to select the number of trees

Boosting

- The first tree predicts the response variable $\hat{y}_i = \hat{f}_1(x_i)$
- The next tree predicts the residuals $r_i^{(0)} = y_i - \hat{y}_i, \quad \hat{r}_i^{(0)} = \hat{f}_2(x_i)$
- The overall prediction becomes $\hat{y}_i^{(1)} = \hat{f}_1(x_i) + \lambda \hat{f}_2(x_i)$ (where λ is a small shrinkage parameter)
- The bth tree predicts the residuals $r_i^{(b)} = y_i - \hat{y}_i^{(b-1)}, \quad \hat{r}_i^{(b)} = \hat{f}_b(x_i)$
- The final prediction (after B trees) is $\hat{y}_i^{(B)} = \hat{f}_1(x_i) + \lambda \sum_{b=2}^{B} \hat{f}_b(x_i)$

Boosting: code

- Jupyter notebook about regression trees
- (On GitHub https://github.com/tedinburgh/ads2023 and the course GitLab)

AdaBoost, gradient boosting, XGBoost

- Boosting corrects previous errors using mean squared error as loss function

- AdaBoost re-computes the weights of each tree at every iteration (works best for binary classification)
- Gradient boosting generalises the MSE to other loss functions
- XGBoost (eXtreme Gradient Boosting) became very popular in 2010s and was often used by winning teams in ML competitions
  - It uses Newton-Raphson method for optimisation, plus better regularisation and various tuned parameters, in order to improve the boosting

Format of the data

$$X = \begin{pmatrix} X_1 & X_2 & \dots & X_p \end{pmatrix} = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \qquad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

- What if features are not independent?
- Rotation forest: for each tree, PCA on a random subset of features

Summary

- Decision trees segment the predictor space, they are interpretable but generally not brilliant at prediction
- Grow a large tree, then prune it back to avoid overfitting
- Ensembles combine lots of trees and are much better at prediction
- Bagging: independent trees from bootstrap samples
- Random forest: restrict the predictors available at each partition
- Boosting: grow successive trees using slow weak learners (shrinking)

Questions?

- Feel free to email me at te269@cam.ac.uk

Next time

- Unsupervised learning
  - Dimensionality reduction
    - Principal component analysis
    - Outliers/anomalies