Lecture 11

Trees and Forests

ISLR 8

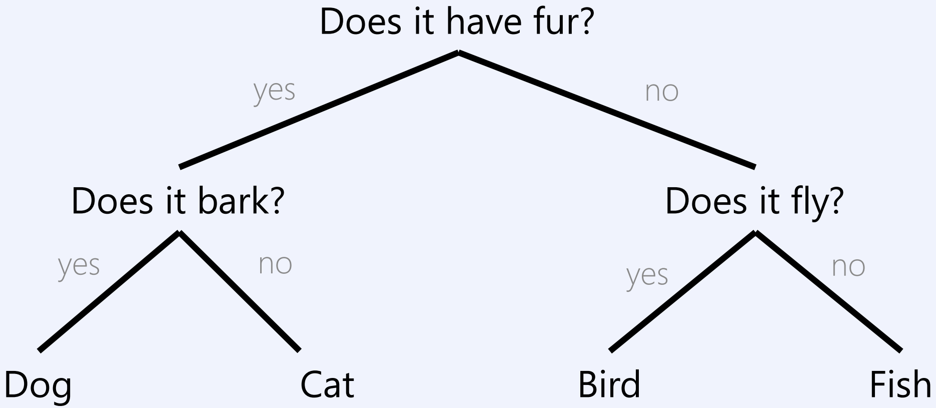
Jilles Vreeken
Krikamol Muandet

UNIVERSITÄT DES SAARLANDES

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

The 20-Questions Game

Intuition for the idea of decision trees: keep asking questions until you figure out the answer



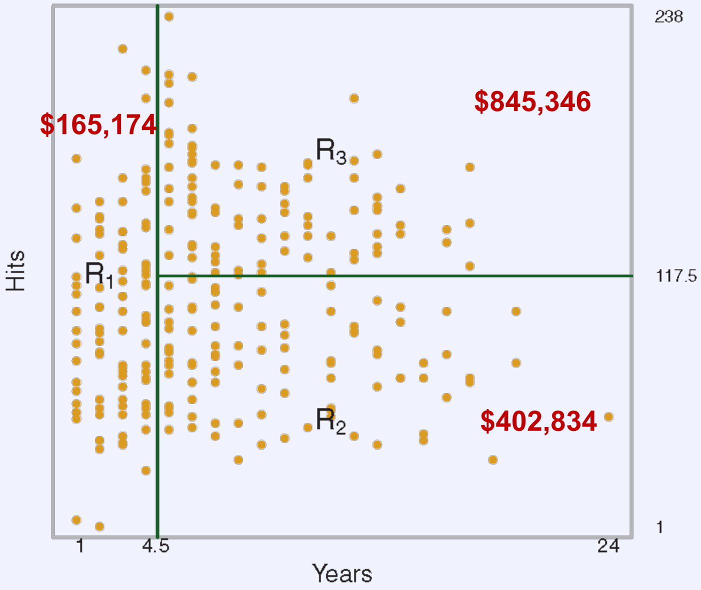
The Basics of Decision Trees

Trees are based on a decomposition of the data space into regions

- a constant model is fit in each region

Example Hitters dataset

- predict salaries of baseball players based on several features
- years #years the player has played
- hits #hits he has made in the previous year
- observations with missing salary data removed first
- then salary scale (in K$) is log-transformed (to the basis e)



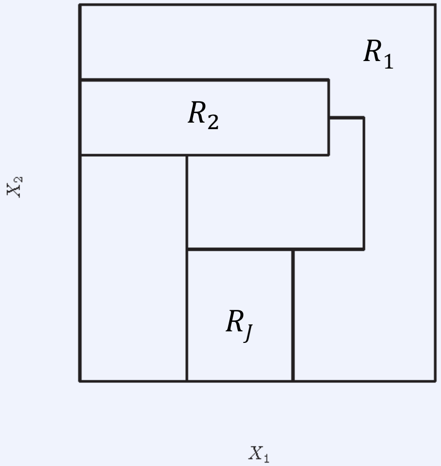
(ISLR 8.1)

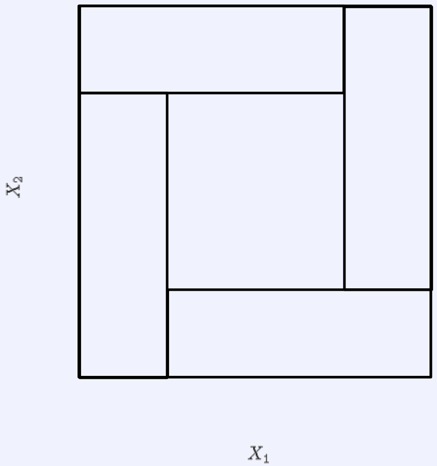
How to Build a Regression Tree

Two simple steps

1. divide predictor space (data space) into J disjoint regions $R_1, \dots, R_J$
2. build a constant model within each region – the mean value of all points in the region

In theory regions can have any shape in step 1, we will only use rectangles (cuboids)

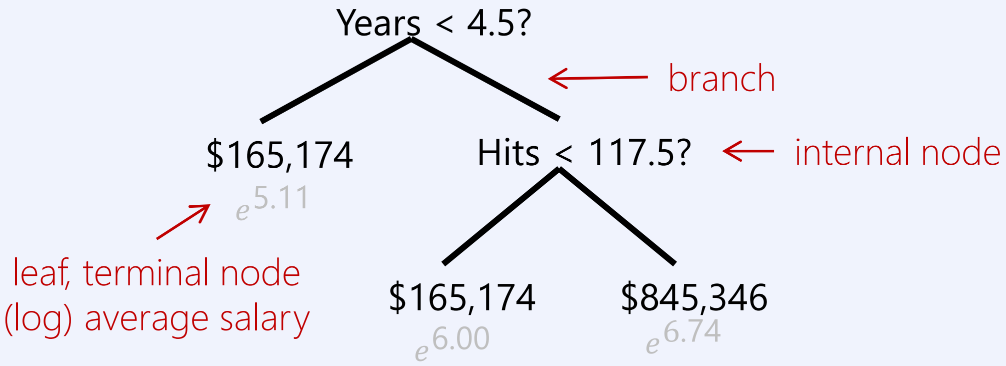




(ISLR 8.1.1)

The Basics of Decision Trees

Trees are easy to interpret and have a nice graphical representation



Years < 4.5?

branch

$165,174

$e^{5.11}$

Hits < 117.5?

internal node

leaf, terminal node
(log) average salary

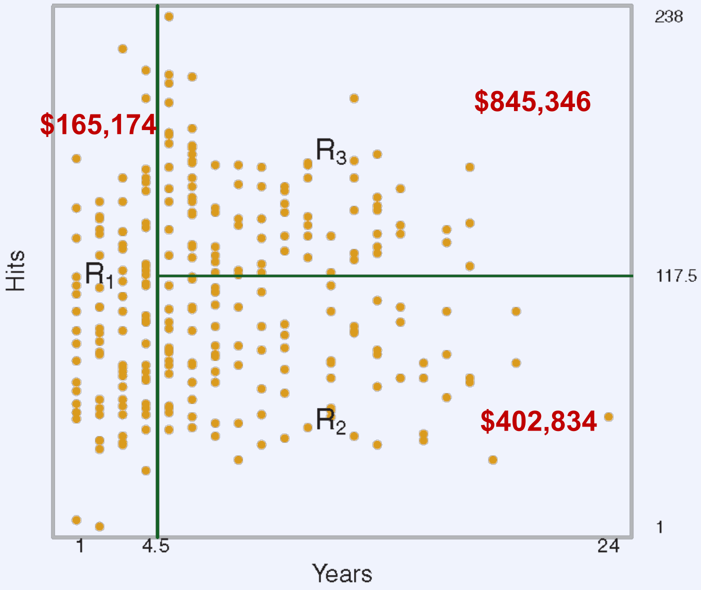
$165,174

$e^{6.00}$

$845,346

$e^{6.74}$

- internal node = feature test, leads to a decision boundary
- branch = different outcome of the preceding feature test
  - if true go left, else if false go right (arbitrary choice)
- leaf = region in the input space and corresponding prediction



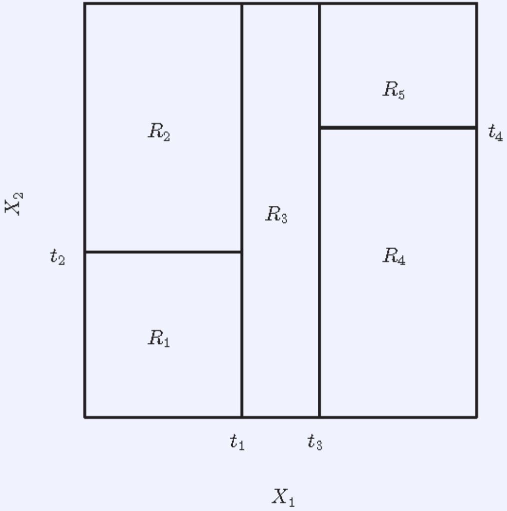
How to Build a Regression Tree

Two steps

1. divide predictor space (data space) into J disjoint regions $R_1, \dots, R_J$
2. build a constant model within each region – the mean value of all points in the region

How to do step 1?

- use rectangles that form Guillotine cuts – cuts that arise from binary splitting the predictor space
- we want to minimize the training error $\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2$



Finding the optimal tree is computationally infeasible (NP complete)

- we follow a greedy strategy: find the currently optimal split at each step

(Hyafil & Rivest. Constructing optimal binary decision trees is NP-complete." Inform Proc Lett 5(1): 15-17, 1976)

How to Build a Regression Tree

Recursively split any leaf such that we minimize the training error

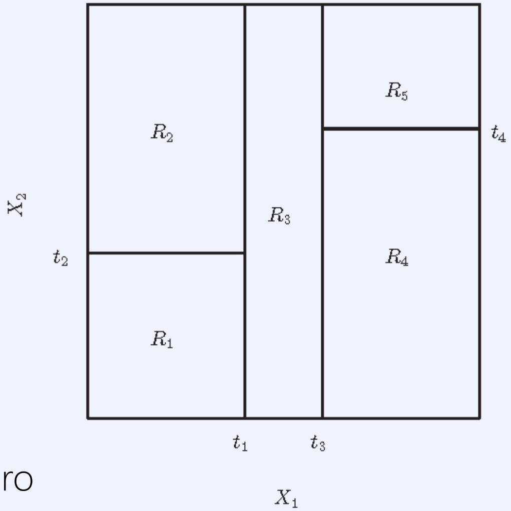
- choose a predictor X_j
- choose a split point s resulting in regions $R_1(j,s) = \{X \mid X_j < s\}$ and $R_2(j,s) = \{X \mid X_j \geq s\}$

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

- as the training data set is finite there are only a finite number of choices
- there are $p \times (n-1)$ choices
- splits are placed half-way between training points

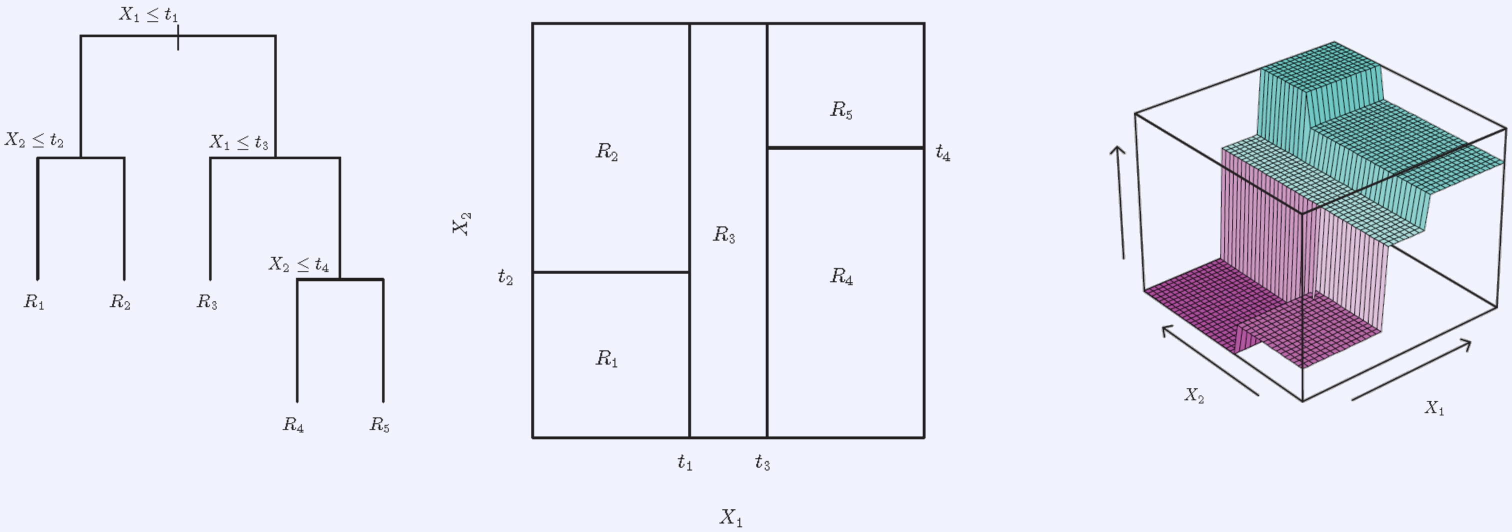
Terminate subdividing regions when some stopping criterion is met

- e.g. when no region has more than five training data points in it
- in principle we can go to one point per region, driving training error to zero



(ISLR 8.1.1)

From Tree to Regions and Back



(ISLR 8.1.1)

Pruning a Tree

Fully grown trees are overtrained (they overfit). Two ways to counter overtraining

1. grow the tree incompletely, or
2. grow the tree fully and then prune it

Variant 1 is suboptimal because we could miss a good cut further down

- greedy procedure lacks foresight

How to find the best pruned tree?

- we want to optimize test error
- we need a complexity criterion assessing test error, i.e. penalizing complex models

(ISLR 8.1.1)

Pruning a Tree

The criterion is formed in analogy to the lasso procedure from Ch. 6

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

penalty parameter α

subtree $T \subset T_0$ of the full tree T_0

- $|T|$ is the number of leaves of T
- $2|T| - 1$ the number of nodes in T
- α controls the tradeoff between fit and complexity
  - $\alpha = 0$ selects the full tree
  - as α increases, the tree gets smaller
- as we increase α the tree shrinks in a nested and predictable fashion

Pruning a Tree

ALGORITHM 8.1 **Building regression trees**

```
1. grow a full tree using recursive binary splitting
2. apply cost complexity pruning to obtain a sequence of best subtrees as a function of $\alpha$
3. use $K$-fold cross validation to choose $\alpha$. For each $k = 1, \dots, K$
    a) repeat steps 1, 2 on all but the $k$-th fold
    b) evaluate the MSE on the left-out $k$-th fold
    average results for each value of $\alpha$ and pick $\alpha$ to minimize average error (or use 1-standard-error rule)
4. return the subtree from step 2 for the chosen value of $\alpha$
```

(ISLR 8.1.1)

Pruning a Tree

ALGORITHM 8.1 **Building regression trees**

1. grow a full tree using recursive binary splitting
2. apply cost complexity pruning to obtain a sequence of best subtrees as a function of α
3. use K-fold cross validation to choose α. For each $k = 1, \ldots, K$

   a) repeat steps 1, 2 on all but the k-th fold

   b) evaluate the MSE on the left-out k-th fold

   average results for each value of α and pick α to minimize average error (or use 1-standard-error rule)
4. return the subtree from step 2 for the chosen value of α

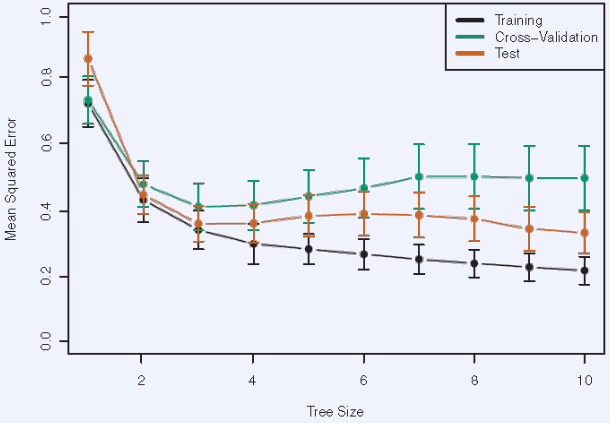
In Step 2 we find a best subtree for each α by weakest-link pruning

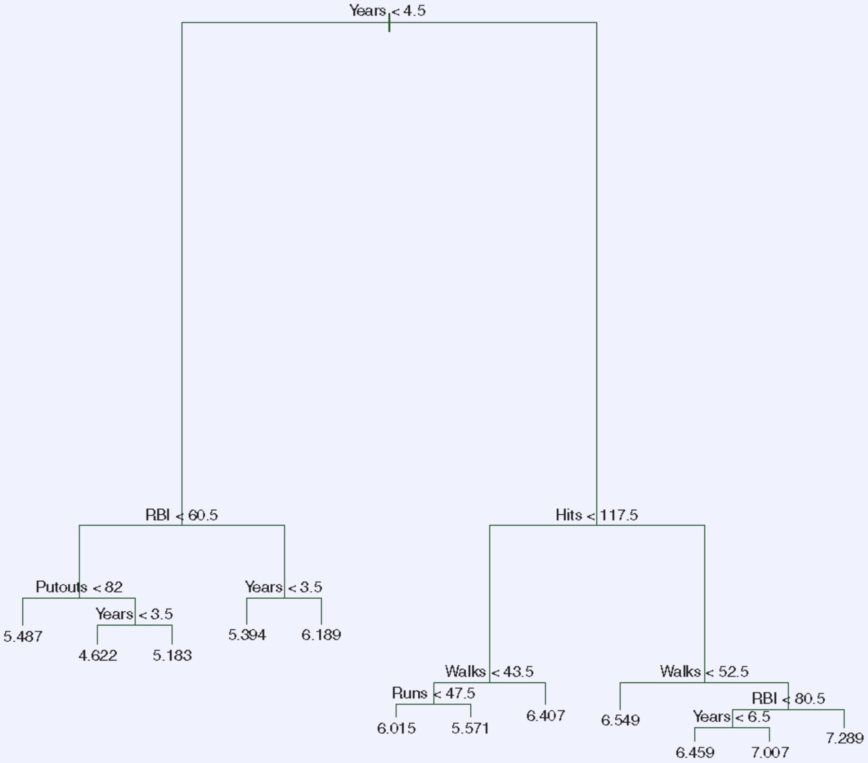
- successively collapse the internal node that causes the smallest increase in the training error
- do so until you get the 1-node tree
- the resulting sequence contains the best subtrees for a in terms of the cost-complexity criterion, for all a

(Breiman, et. al. Classification and Regression (1984)), (Ripley, Pattern Recognition and Neural Networks, (1996)), (ESL 9.2.2)

Example Regression Tree on the **`Hitters`** Dataset

- nine features
- 132 training, 131 test points
- six-fold CV on the training set
- evaluation on the test set





Classification Trees

Like a regression tree prediction inside a region is the modal class of that region

- the majority class – class with most members in the region (Bayes classifier)
- class proportions inside the region are interesting for interpretation

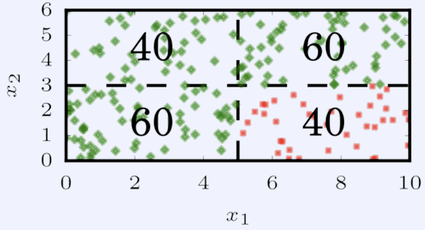
Idea for tree growing: Use the misclassification error as a loss function instead of square error

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

- $\hat{p}_{mk}$ is the fraction of points in region R_m that belong to class k

The misclassification error is not sufficiently sensitive. For example before split $E = \frac{40}{200}$

- best split on $x_1 < 5$ is , $E_{\text{left}} = \frac{0}{100}$, $E_{\text{right}} = \frac{40}{100}$ so the average error after is $0.5E_{\text{left}} + 0.5\ E_{\text{right}} = \frac{40}{200}$
- similar thing holds for the best split on $x_2 < 5$



(ISLR 8.1.2)

Other Suitable Criteria (Impurity Measures)

Desiderata: maximum if classes are equally distributed and minimum (usually 0) if node is pure

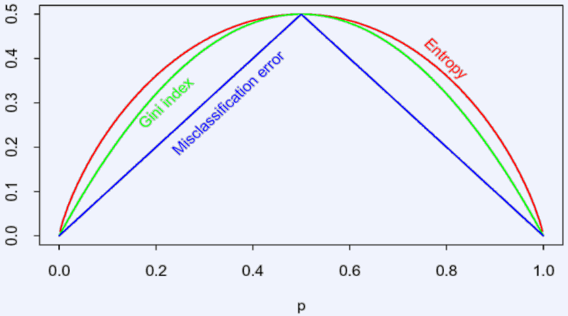
The Gini index is a measure of total variance across the k classes $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$

- is small if all $\hat{p}_{mk}$ are close to zero or one
- is the average training error if in region R_m we classify a point to class k with probability $\hat{p}_{mk}$
- code the response as 1 for class k and as zero for all other classes. The variance over this response in region R_m is $\hat{p}_{mk}(1 - \hat{p}_{mk})$, summing over all classes gives the Gini index

Another loss function is cross-entropy $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

- since $0 \leq \hat{p}_{mk} \leq 1$ we have $0 \leq -\hat{p}_{mk} \log \hat{p}_{mk}$
- the cross-entropy is near zero if all $\hat{p}_{mk}$ are near zero or one.

Usually we use G and D for tree building and E for pruning

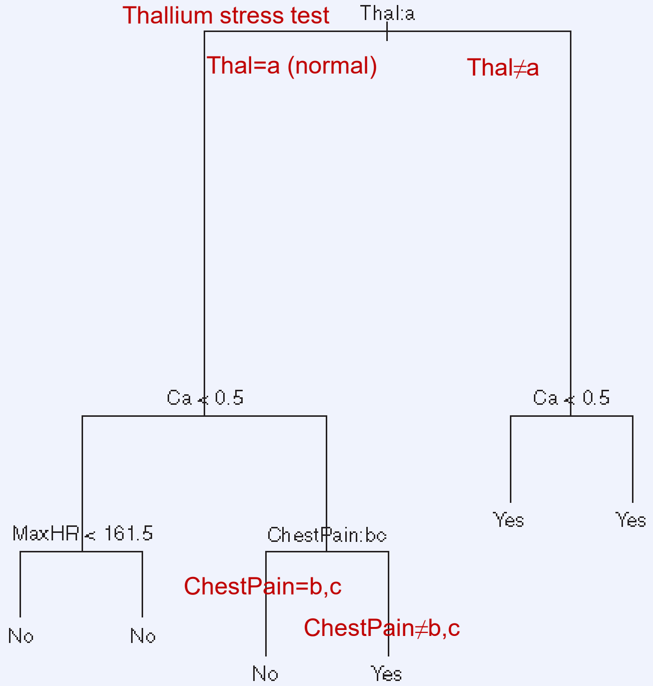


impurity measures (binary classification)

(ISLR 8.1.2)

Example Classification Tree on **Heart** Data

- 303 observations of patients with chest pain
- **HD** response, binary: heart disease based on angiographic test with outcome **Yes** or **No**
- 13 predictors, including **Age**, **Sex**, **Chol**
- CV results in tree with six nodes



pruned tree

(ISLR 8.1.2)

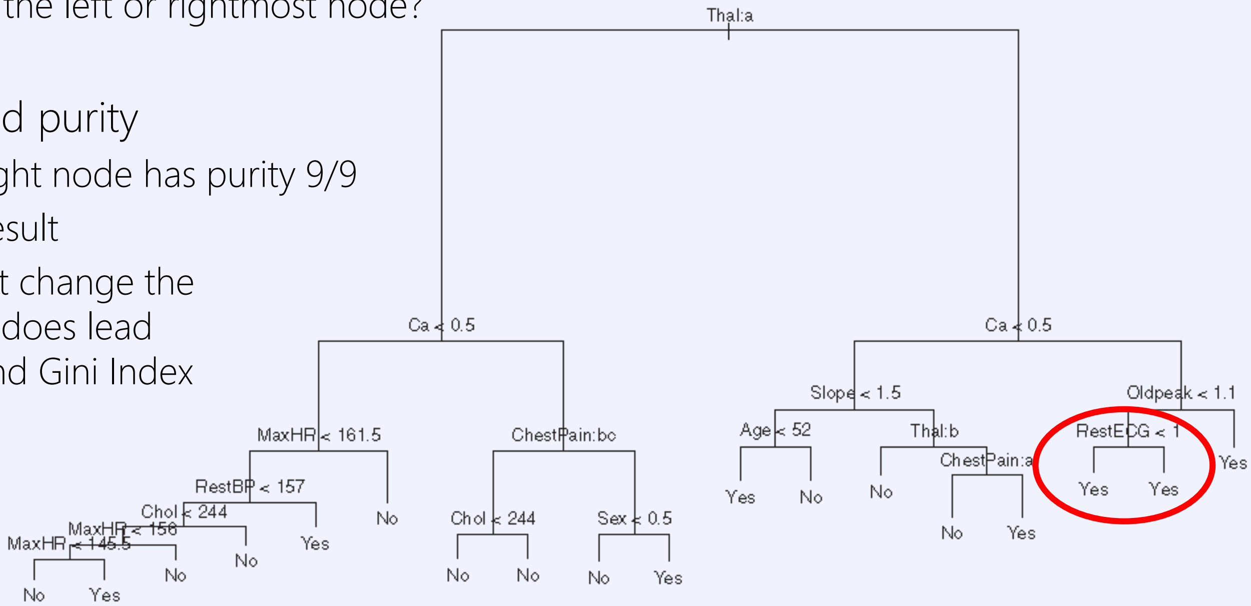
Example Classification Tree on **Heart** Data

The unpruned tree

- why is a split performed on the left or rightmost node?

Because it leads to increased purity

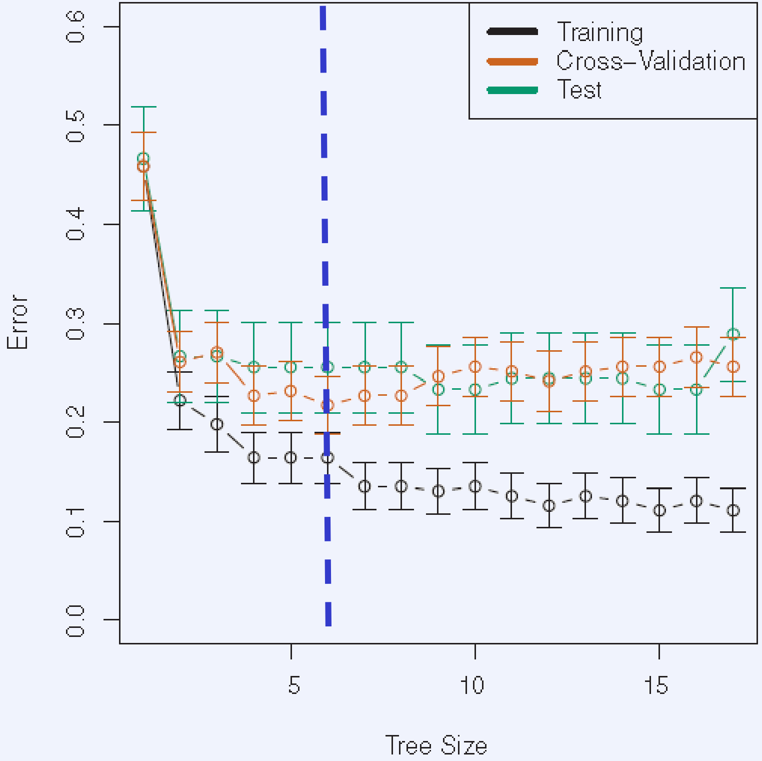
- left node has purity 7/11, right node has purity 9/9
- related to certainty of the result
- splitting the region does not change the misclassification error but it does lead to reduced cross entropy and Gini Index



(ISLR 8.1.2)

Example Classification Tree on **Heart** Data

- 303 observations of patients with chest pain
- **HD** response, binary: heart disease based on angiographic test with outcome **Yes** or **No**
- 13 predictors, including **Age**, **Sex**, **Chol**
- CV results in tree with six nodes
- exact configuration of CV and test not described in the book



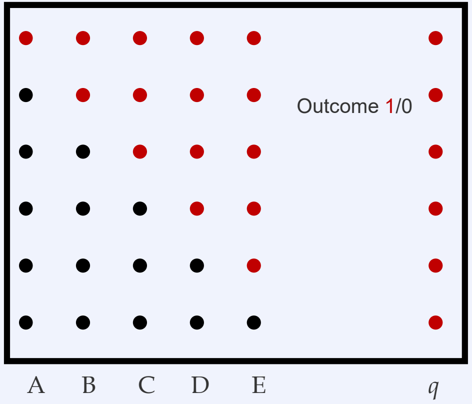
(ISLR 8.1.2)

How to Split Categorical Predictors

A predictor with q unordered values allows $2^{q-1} - 1$ splits, optimization is prohibitively costly

However, if we have a task with a 0-1 outcome allows for simpler computation

- order all categories of predictor according to the fraction of observations falling into output class 1
- then split the predictor as if it were a quantitative (ordered) predictor
- this optimizes the split in terms of both Gini index and cross entropy
- same holds if outcome is quantitative as we can order by increasing mean of the outcome



(Breiman, et. al. Classification and Regression (1984)), (ESL 9.2.4)

Missing Predictor Values

We could fill missing values (e.g. with mean of observations)

- for tree-based methods, we can do better

Categorical predictors: make a new category for “missing”

General approach: introduce surrogate variables

- define best (primary) predictor and split point as usual (based only on existing observations)
- form a list of surrogate predictors
- first surrogate is the predictor (and split point) that best mimics the split by the primary predictor
  - finding such a predictor is involved but decision tree software offers the feature
- second surrogate is second-best, etc.
- during prediction, if the predictor value is missing, use the first surrogate for which a value is available
- the higher the correlation between predictors the better this works

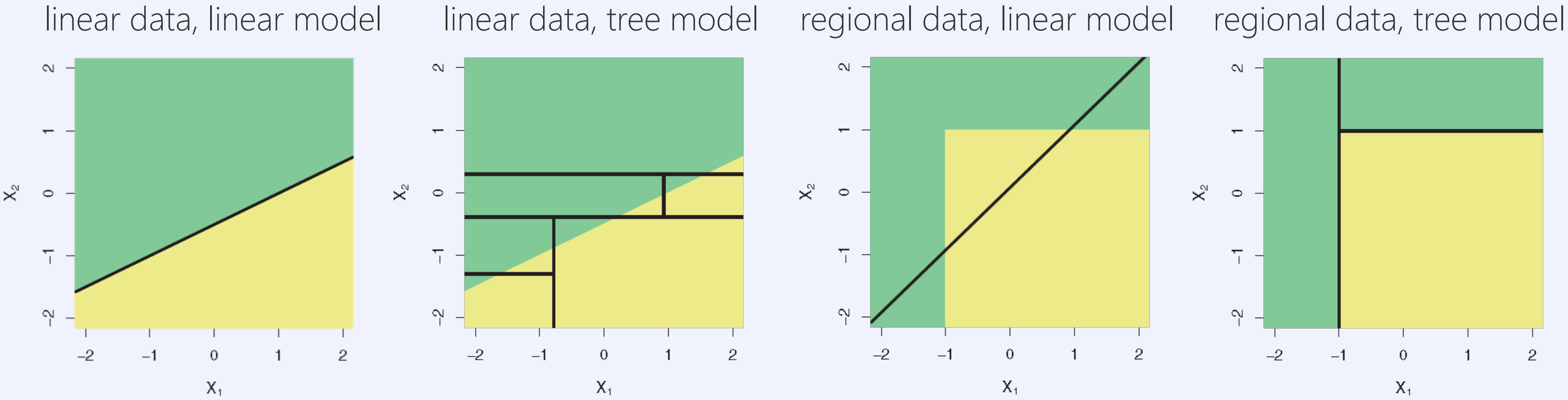
(Rokach & Maimon. Decision Trees. The Data Mining and Knowledge Discovery Handbook. 149-174, Springer, 2010) , (ESL 9.2.4)

Trees vs. Linear Regression

Linear regression $f(X) = \beta_0 + \sum_{j=1}^{p} \beta_j X_j$, the world is globally linear

Trees $f(X) = \beta_0 + \sum_{m=1}^{M} c_m I(X \in R_m)$, the world is regionally constant

Which model is more suitable depends on the problem, example 2D binary classification



linear data, linear model

linear data, tree model

regional data, linear model

regional data, tree model

(ISRL 8.1.3)

Advantages and Disadvantages of Trees

- ➕ Trees are easy to explain to people
- ➕ Trees arguably mimic human decision-making
- ➕ Trees have a simple graphical representation and are easy to interpret, especially, if they are small
- ➕ Trees can handle qualitative predictors without the need of dummy variables
- ➕ Trees allow for systematically imputing missing values

- ➖ Trees are often not as accurate as the other models
- ➖ Trees can be very non-robust, i.e. performance can change dramatically upon small changes in the data

(ISLR 8.1.4)

Ensemble Methods based on Trees

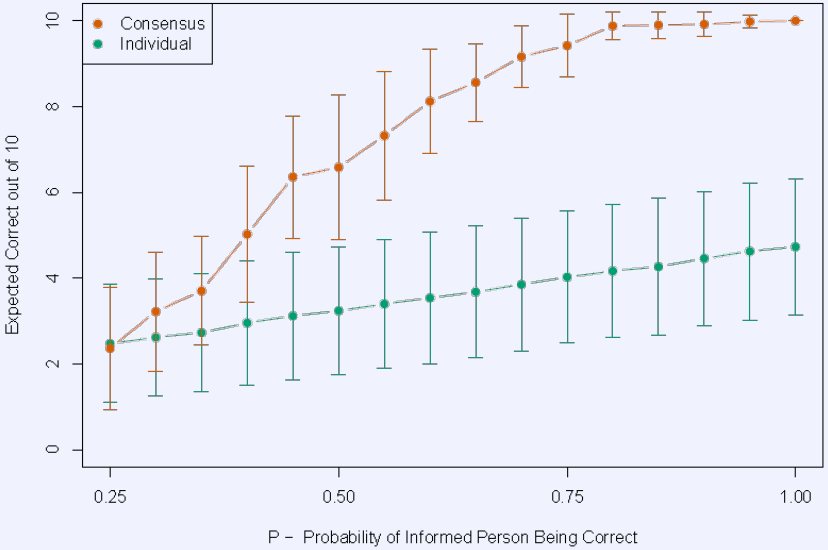
Ensemble methods calculate **several models** for a dataset and **merge their predictions**

- equivalent of getting second and third opinions in daily life
- often getting several predictions can reduce variance
- “Wisdom of the Crowds” phenomenon

Wisdom of the Crowds

Example simulated academy awards voting

- 50 members vote in 10 categories with 4 nominations each
- for any category, only 15 random members have some knowledge of selecting the "correct" candidate ($p = 0.25$ means no knowledge)
- error bars are one standard deviation



(ESL 8.7.1)

Ensemble Methods based on Trees

Ensemble methods calculate several models for a dataset and merge their predictions

- equivalent of getting second and third opinions in daily life
- often getting several predictions can reduce variance
- “Wisdom of the Crowds” phenomenon

We will discuss three ensemble methods

1. Bagging: Merging the tree approach with the bootstrap approach
2. Random forests: Decorrelating bagged trees
3. Boosting: Weighting data points by difficulty and models by accuracy

Bagging

Apply the bootstrap method (Ch 6) to tree models to reduce the variance

1. generate B training datasets using the bootstrap
2. build a tree on each dataset affording the response $\hat{f}^{*b}(x)$
3. average over the response of all trees for the final prediction $\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$

Especially helpful for decision trees (but provably unhelpful for linear least-squares)

- Trees are grown deep, unpruned each individual tree thus has high variance but low bias
- averaging over many trees should reduce the variance

For regression predict the average, for classification predict the majority vote of all trees

(ISLR 8.2.1)

Random Forests

Bagged trees have a problem

- because bootstrap samples have a large overlap, bagged trees are highly correlated
- the decrease of variance by averaging over them is thus not as large as desired
- Recall: the variance of the average of k i.i.d. samples with variance σ and correlation ρ, *is* $\rho\sigma^2 + \frac{1-\rho}{k}\sigma^2$

Trick to decorrelate the trees: instead of the full set of predictors, choose the best predictor out of a random sample of m predictors

- usually m is chosen as $m = \sqrt{p}$ for classification and $m = p/3$ for regression
- works since different trees will typically get to use different predictors

Example: assume there is one very strong predictor

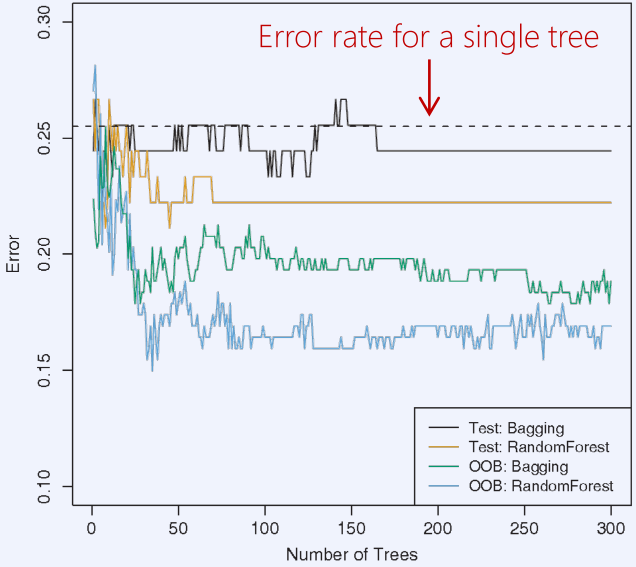
- in bagging, all trees will have this predictor, probably even at the top, thus they are very similar
- the random forest will have the predictor only in m/p of the splits

(Breiman. Random Forests. Machine Learning 45(1):5-32, Springer, 2001)

Example Bagging vs. Forests on the **Heart** Data

Test error of bagged trees can be calculated on out-of-bag sample

- recall that on average about 1/3 of a bagged sample is out of bag
- thus each observation will be out of bag for about 1/3 of the trees
- we can use those trees to assess the test error on that observation
  - for the top two curves data are randomly split in training and test set
  - the relative smoothness of the curves is due to high correlation of the error estimates on similar numbers of trees



Choice of B: Bagging does not overfit with growing $\boldsymbol{B}$

- so choose a B for which the test error has “settled down”

Random forests perform better since it decorrelated the trees

(ISLR 8.2.1)

Choosing the Value of m

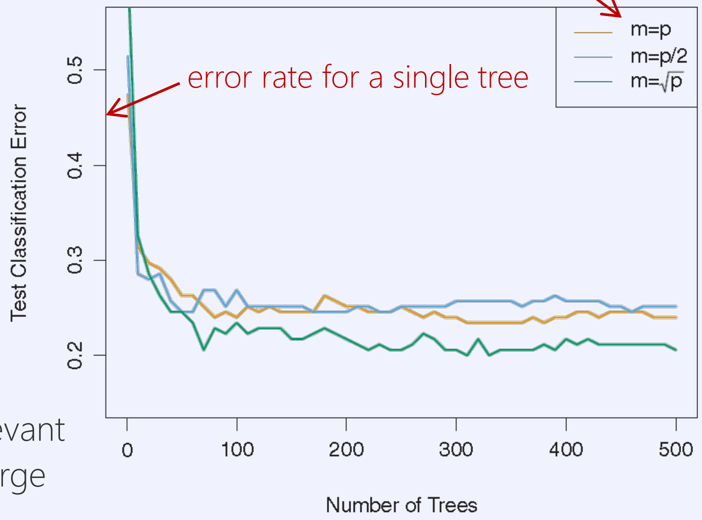
A small value of m like $m = \sqrt{p}$ is appropriate for a large number of correlated predictors

- this typically is the case in biological datasets
- $m = p$ is equivalent to bagging

Example cancer type prediction

- high-dimensional dataset with expression levels of 4,718 genes for 349 patients of 14 different cancer types + control
- goal: predict cancer type (15 classes) based on the expression levels of 500 genes with the largest variance in expression in the training set
- test error assessed with the validation set approach
- gain by random forest over bagging is less than expected, probably due to increased bias: we expect only a few genes are relevant
- thus bias can be due to forced inclusion of false genes when m is large

random forests on a cancer dataset



(ISLR 8.2.1)

Variable Importance Measures

How to interpret additive models of trees?

- **key issue:** relative importance of predictor variables

improvement in square error at a split node *t* over that of a constant fit in the entire region

internal tree node

Breiman (1984) proposed the following measure for a single tree $I_l^2(T) = \sum_{t=1}^{J-1} \hat{\imath}_t^2 I(v(t) = l)$

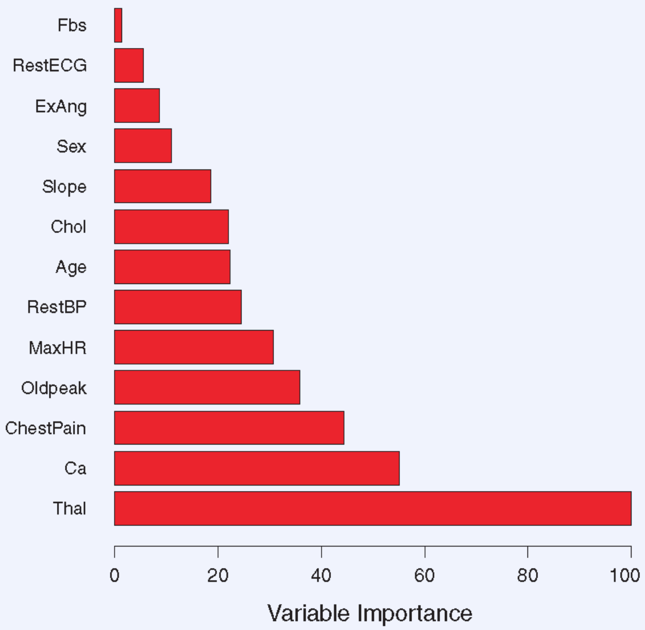
- measures the relevance of predictor variable X_l
- sum is over the internal tree nodes
- in node t the variable $X_{v(t)}$ is used for splitting

Recall: the variable maximally reducing the squared-error is chosen at each tree-growth step

- this improvement is $\hat{\imath}_t^2$
- the squared importance of X_l is the sum over all nodes split by X_l
- generalized to model ensembles by averaging over all trees $I_l^2 = \frac{1}{M} \sum_{m=1}^{M} I_l^2(T_m)$

Example Variable Importance Measures (`Heart`)

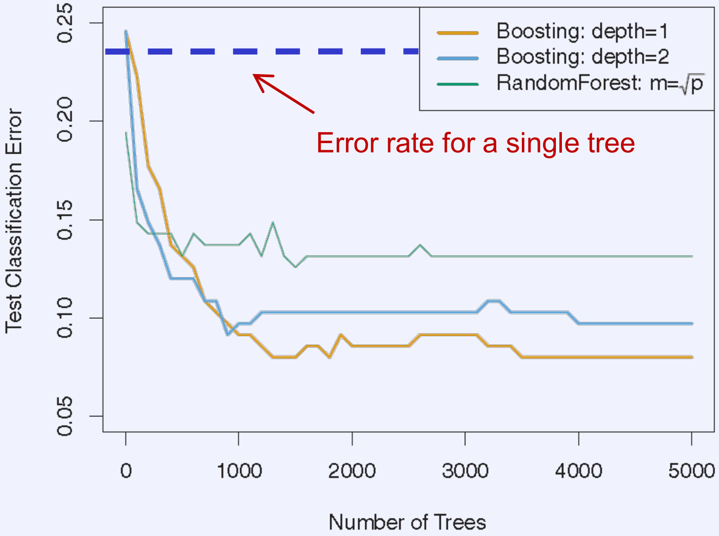
- Gini index is used as loss function
- data scaled such that maximum importance value is 100



Boosting

Boosting is a powerful ensemble technique

- solid theoretical foundation for binary classification
- in contrast to bagging and random forests, here trees are not calculated independently but in sequence
- AdaBoost (ESL 10.1): original version of boosting as applied to binary classification with trees
- we present a heuristic version of boosting that applies to any classification and regression trees



boosting on the cancer dataset
for predicting “cancer” vs. ”normal”
std. errors are about 0.02

(ISLR 8.2.3)

ALGORITHM 8.2: Boosting for Regression Trees

```
                 residual
1.  Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all observations $i$ in the training set
2.  For $b = 1,2,\ldots,B$ repeat
    a)  Fit a tree $\hat{f}^b$ with $d$ splits ($d + 1$ leaves) to the training data + residual $(X, r)$
    b)  Update $\hat{f}$ by adding a shrunken version of the new tree $\hat{f}(x) = \hat{f}(x) + \lambda \hat{f}^b(x)$
    c)  Update the residual $r_i = r_i - \lambda \hat{f}^b(x_i)$ for $i = 1,2,\ldots,n$
3.  Output the boosted model $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$
```

Boosting means learning slowly (generally a good idea in statistical learning)

- learning small trees (d)
- learning shrunken trees (λ), small λ allows more trees in the model
- we slowly improve the model in areas in which the model does not perform well

(ISLR 8.2.3)

Model Parameters of Boosting

Number of trees B

- in contrast to bagging and random forests* boosting can overfit, but does so slowly
- use CV to select B

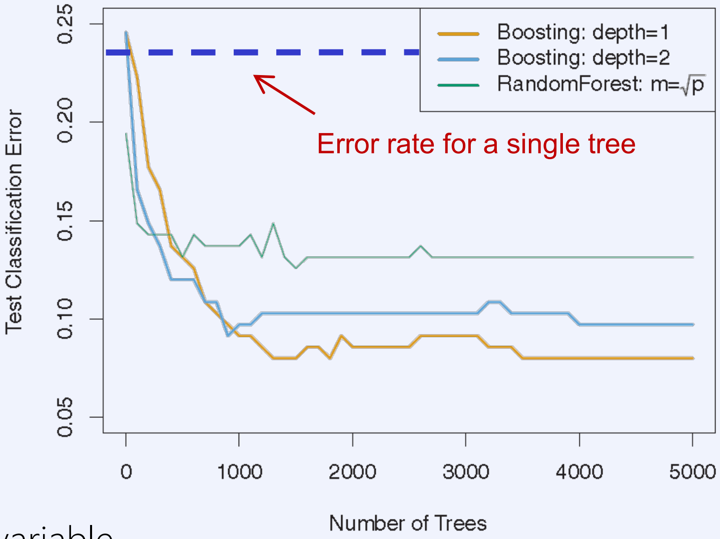
Shrinkage parameter λ

- typical values are 0.01 or 0.001
- small λ require large B

Number of splits d per tree controls complexity

- d=1 (stumps) often works well.
- model is then additive as each tree involves only single variable
- d controls the interaction order of the boosted model

boosting on the cancer dataset for predicting "cancer" vs. "normal" std. errors are about 0.02



Error rate for a single tree

* but see ESL 15.3.4

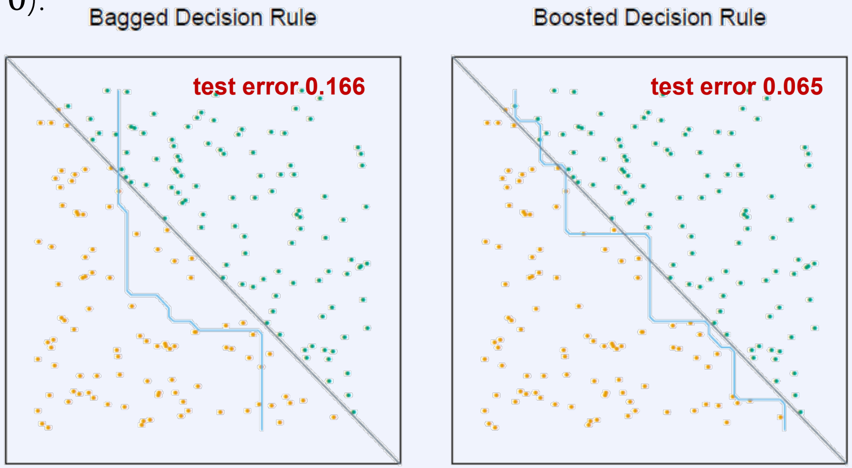
Trees with Simulated Data

Bagging does not always help

- bagging on decision stumps $\hat{G}(x)$ with $N = 100$ and $B = 50$

Dataset generated with $p = 5$ features and 2 classes, separated by $x_1 + x_2 = 1$

- all trees split near the middle of the diagram ($x = 0$).
- averaging is the wrong strategy



Summary

Trees: decompose the space into regions and fit a constant model in each region

- optimal tree is hard, so we reclusively split the data, greedily selecting the current best predictor

Bagging: apply the bootstrap method to tree models to reduce the variance

- because bootstrap samples have a large overlap, bagged trees are highly correlated

Random forests: apply a trick on top of bagging to decorrelate the trees

- randomly sample out of $m < p$ predictors at each split

Boosting: slowly improve the model in areas in which it does not perform well

- in each iteration fit a small and/or shrunken tree on the residuals