Trees and Bagging

Agnes M Nielsen

DTU
Based on Slides by Lars Arvastson and Line Clemmensen

02582 Computational Data Analysis, 2019

Todays Lecture

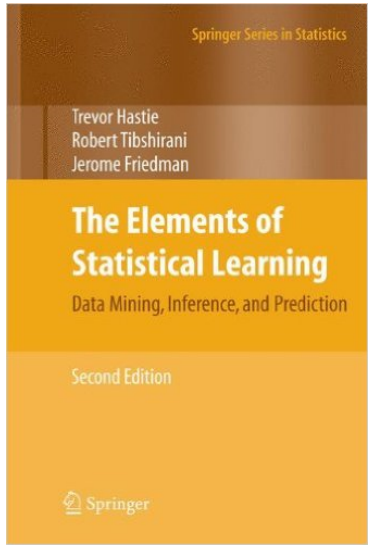
- Recap
- Regression Trees
- Classification Trees
- Bagging

Last Week

- Optimal Separating Hyperplane
- Lagrange Multipliers
- Support Vector Machines
- Kernel Trick

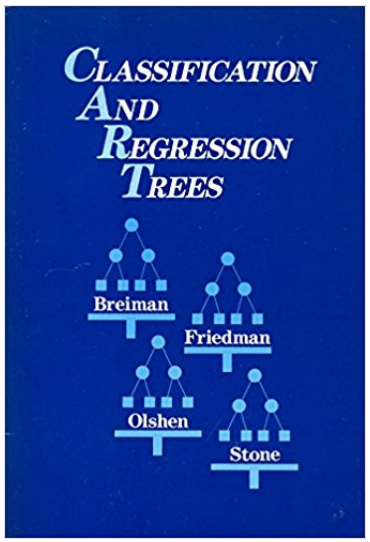
Classification and Regression Trees

Classification **A**nd **R**egression **T**rees (CART)



The Elements of Statistical Learning:

"Tree based methods partition the feature space into a set of rectangles, and then fit a simple model (like a constant) in each one."

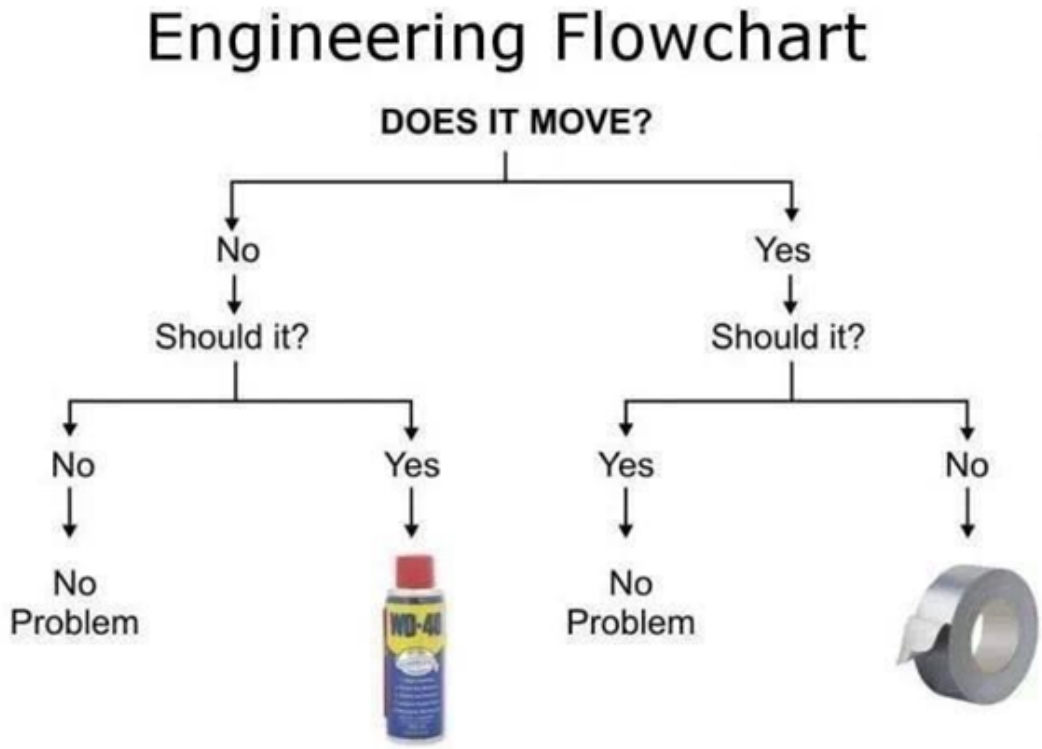


Classification And Regression Trees (1984) by Breiman, Friedman, Olshen and Stone introduced,

- CART
- The one-standard-error rule

Decision Trees

Like a decision tree but built on data instead of expert knowledge



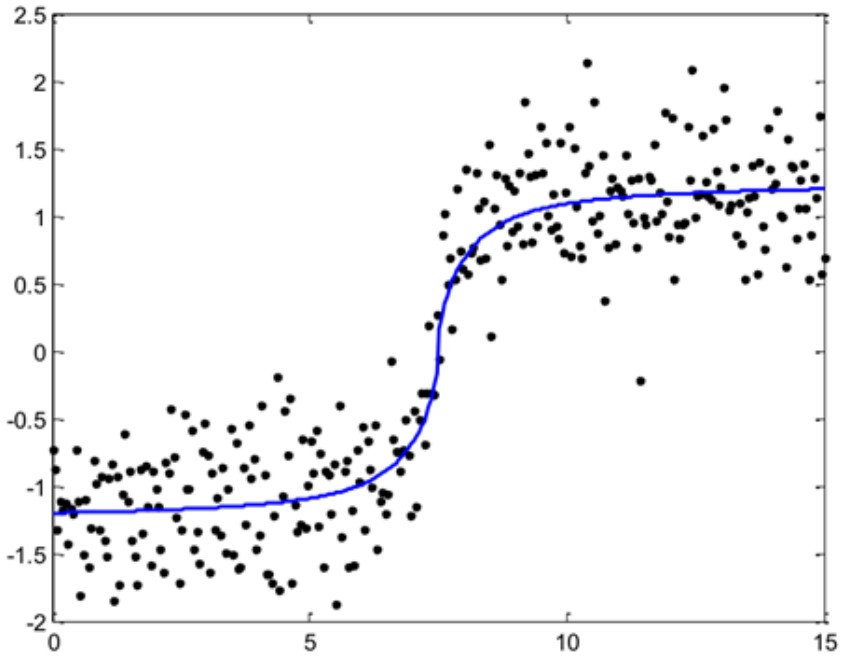
Regression Trees

- Regression Method

A Simple Example

Regression

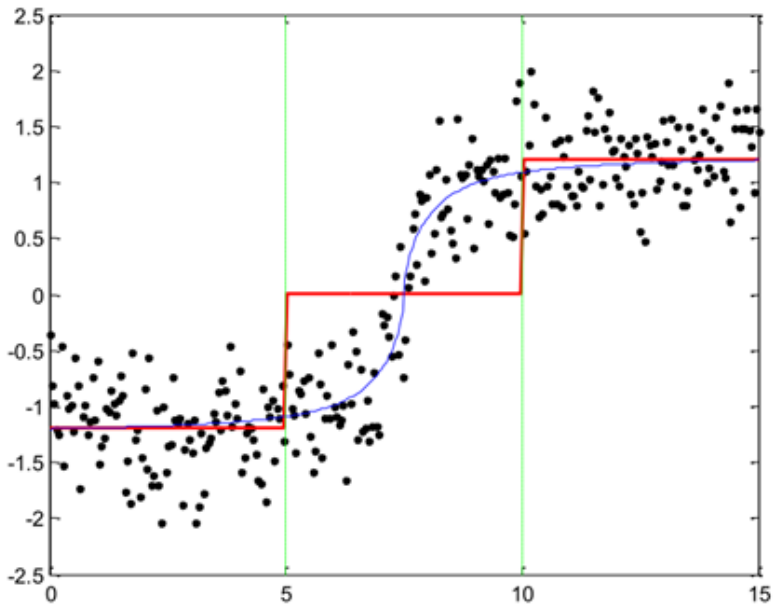
- True function: $y = f(x)$ (blue)
- Observation with noise: (x_i, y_i) (black)

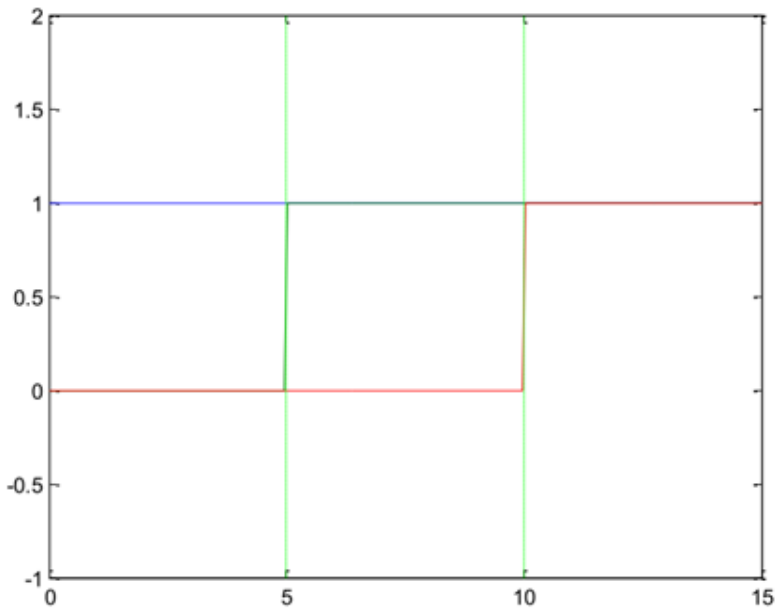


A Simple Example

Fit a piecewise polynomial

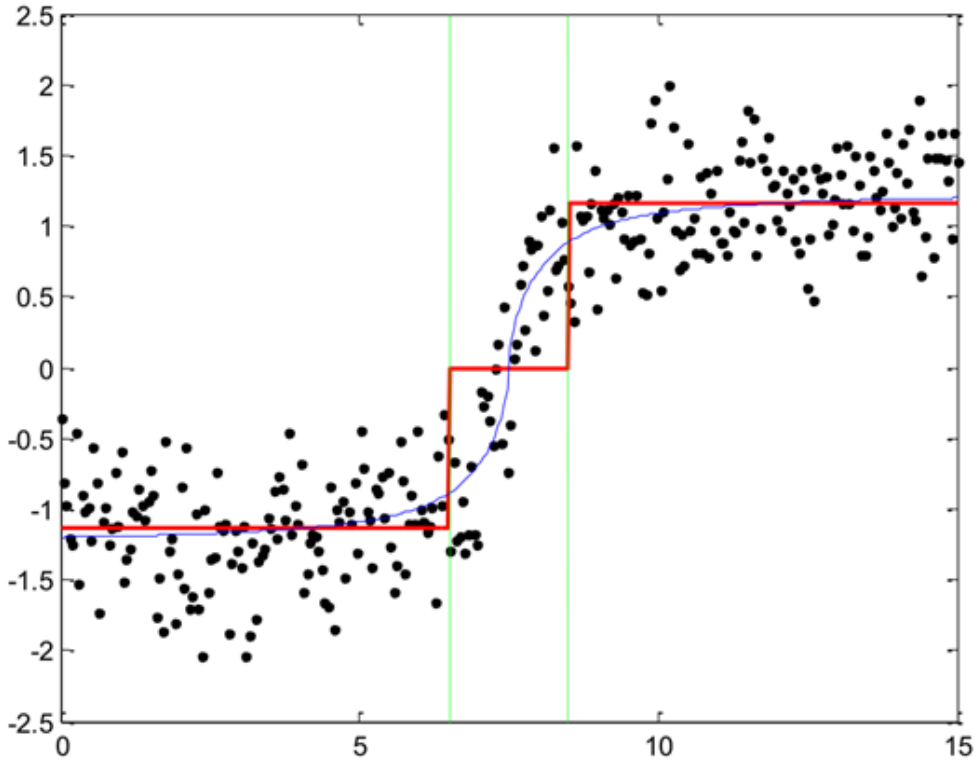
- `X = [ones(n,1) double(x>5) double(x>10)];`
- A constant in each interval
- Evenly spaced knots





A Simple Example

Can we place the knots in a better way?



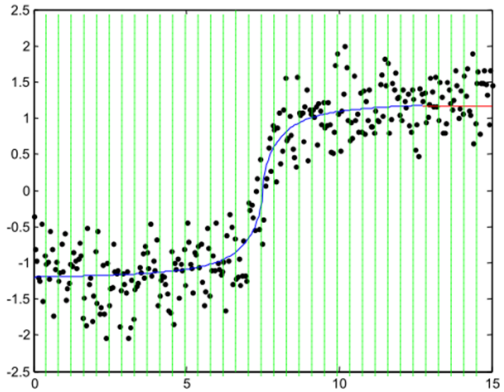
A Simple Example

Algorithm attempt:

- Choose a number of knots k.
- try all possible positions for each knot k
  - Infinite number of combinations

Try e.g. 100 positions on the x-axis for each knot

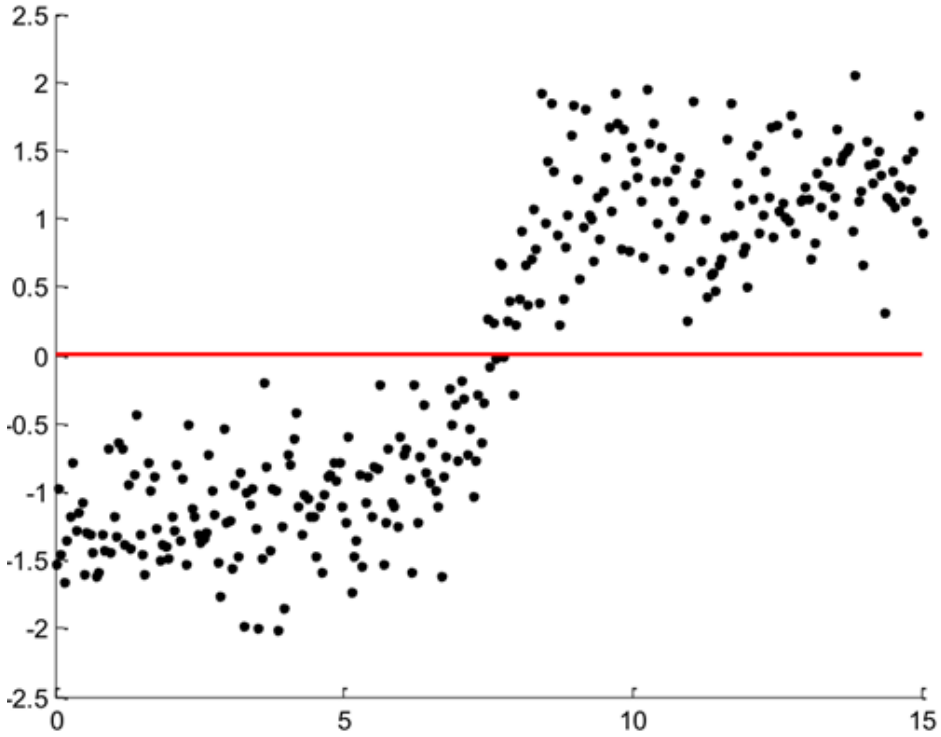
- 100^k positions to try
- E.g. $100^5 = 10\ 000\ 000\ 000$ combinations
- With more than one input variable it gets even worse



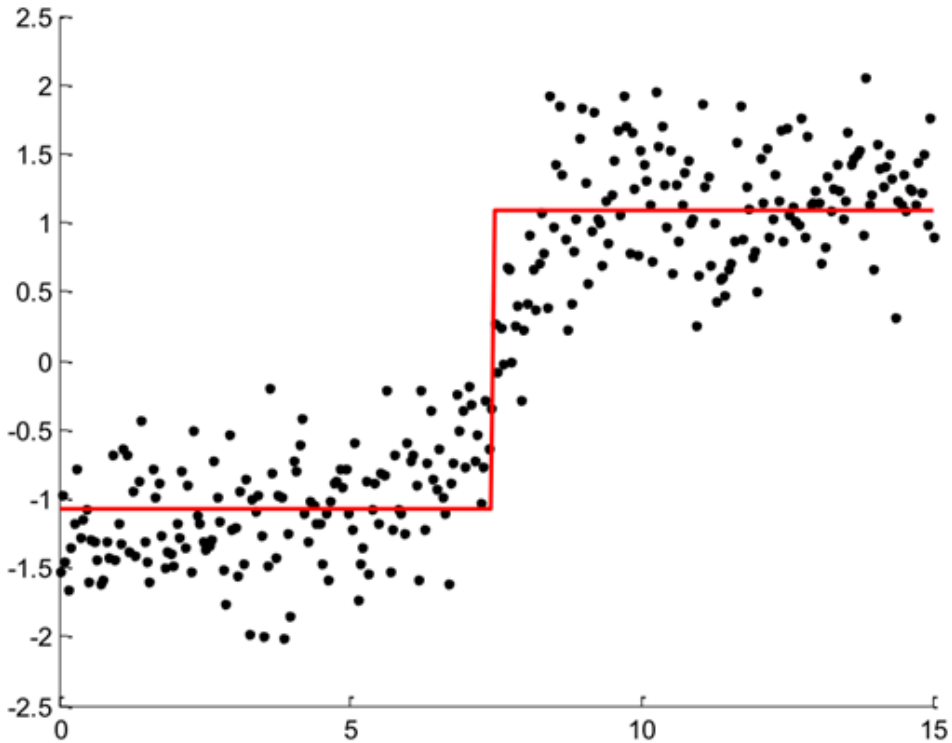
A Simple Example

New idea:

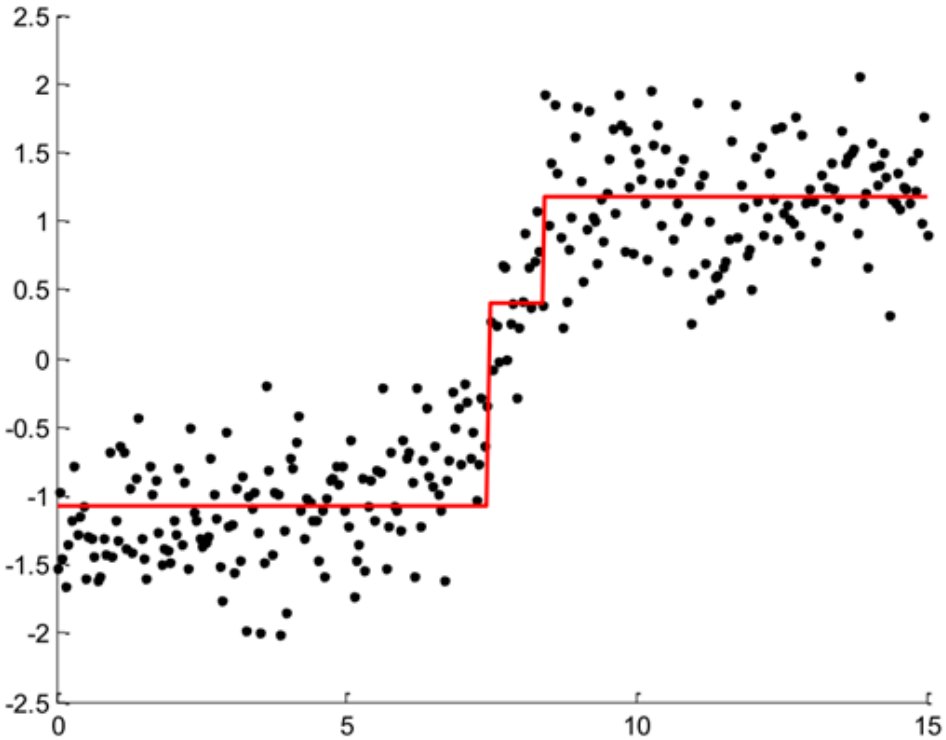
- Place the knots one after another
- Place each knot such that the fit is as good as possible



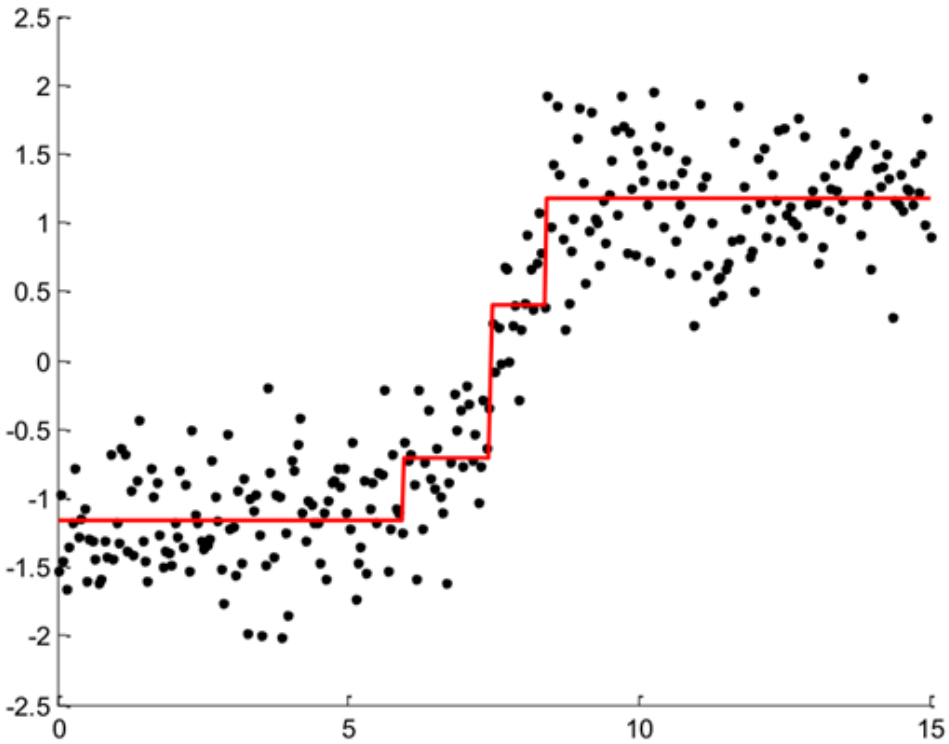
A Simple Example



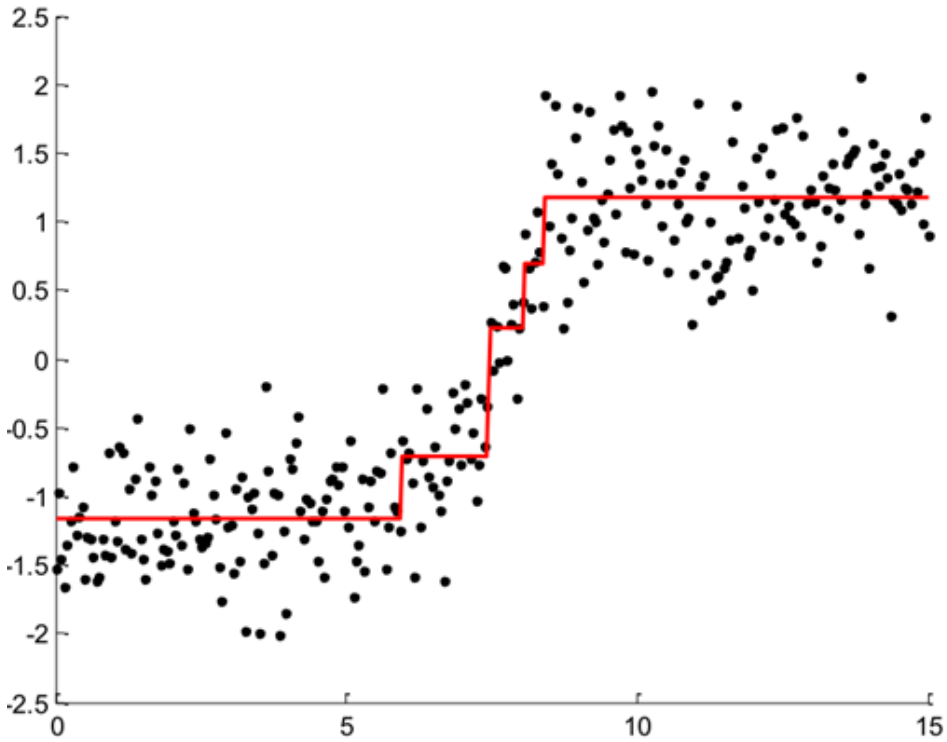
A Simple Example



A Simple Example

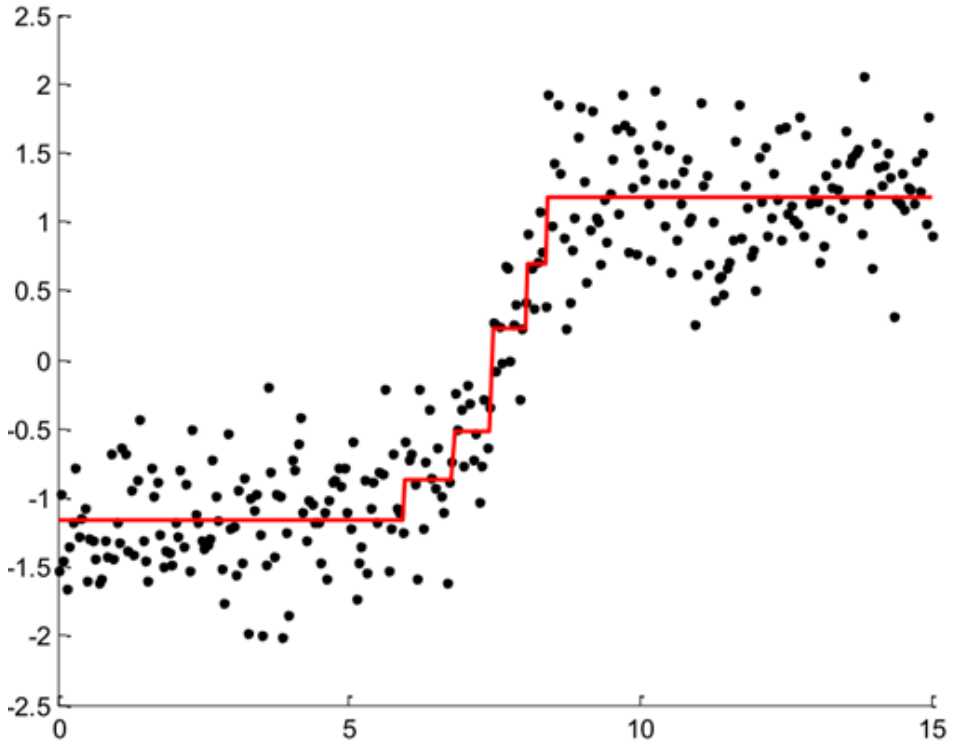


A Simple Example



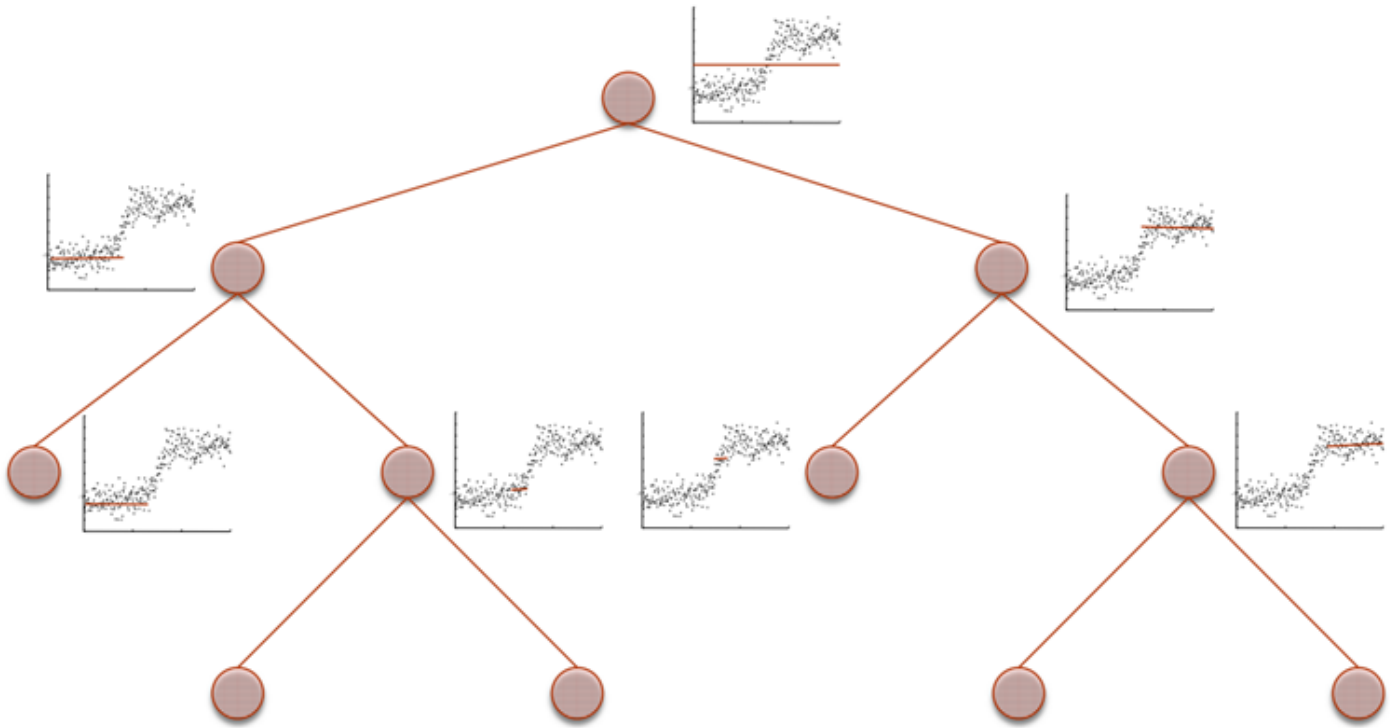
A Simple Example

At this point, the regression function is pretty good!



Tree Representation

Each split can be represented by a parent node split into two child nodes



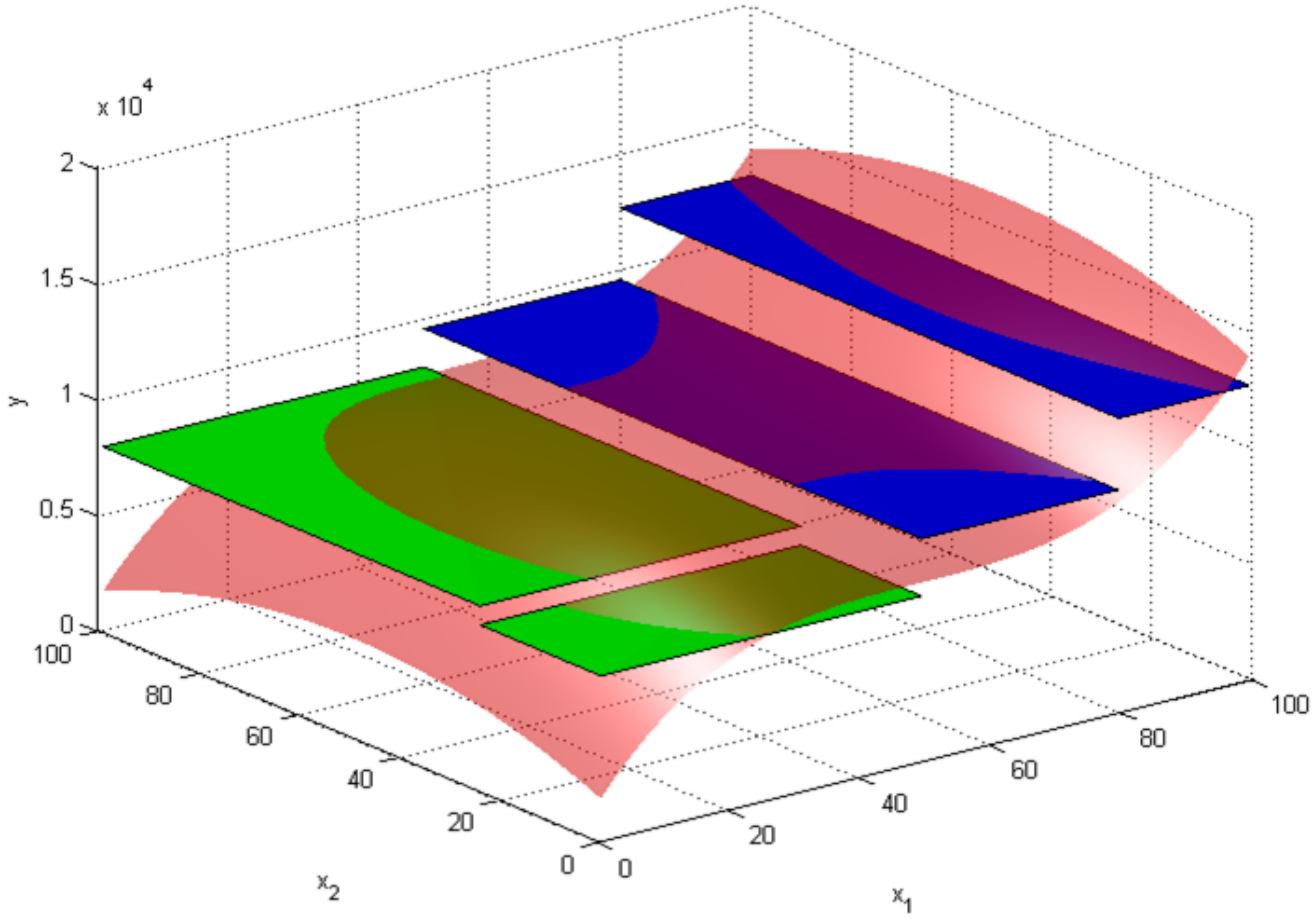
More Than One Input Variable

Handled in the simplest way possible

For each input variable
    For each split
        Evaluate split point
    End
End

Choose the best variable and split

More Than One Input Variable



Questions

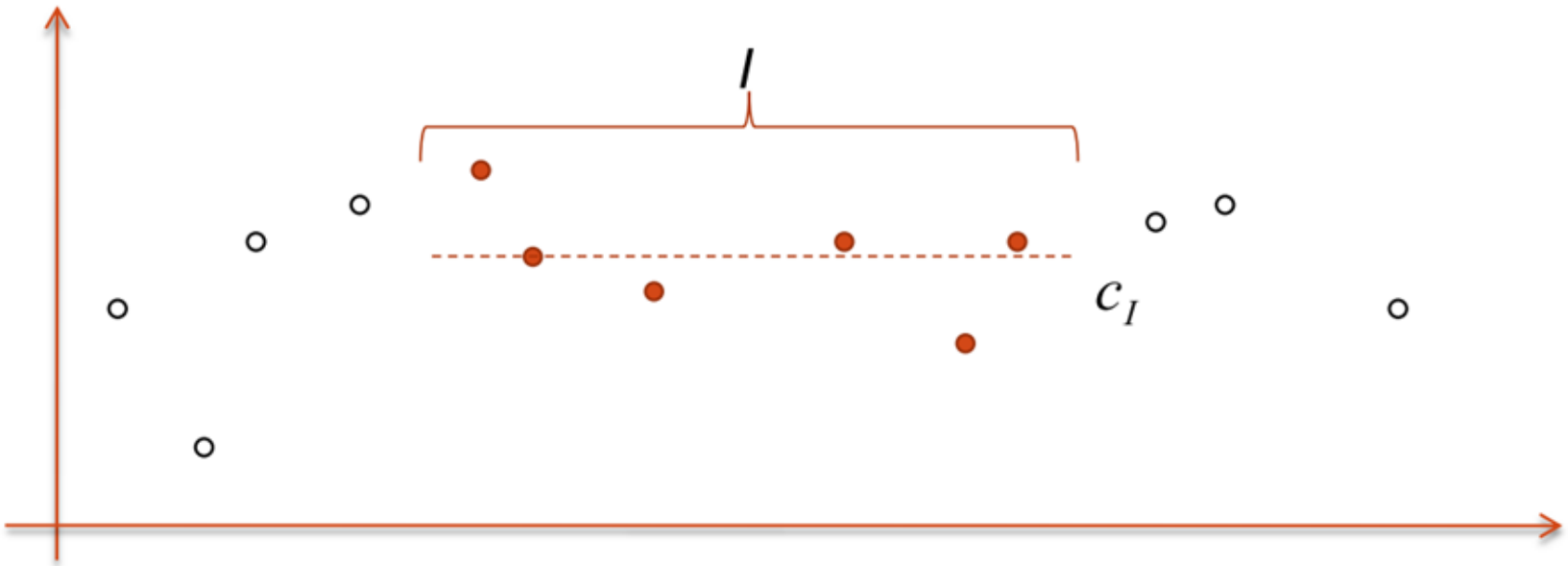
- What is a good split?
  - We need to know this to decide where to split
- How many splits should we try?
  - We used 100 before - is there a better choice?
- When do we stop splitting?

What is a Good Split?

- The terminal nodes each represent an interval of the input variable(s).
- We choose to represent the outcome in this interval by a **constant** function.
- As for most regression problems, we say that a constant function is good if it has low **residual sum of squares** when compared to training outcome data

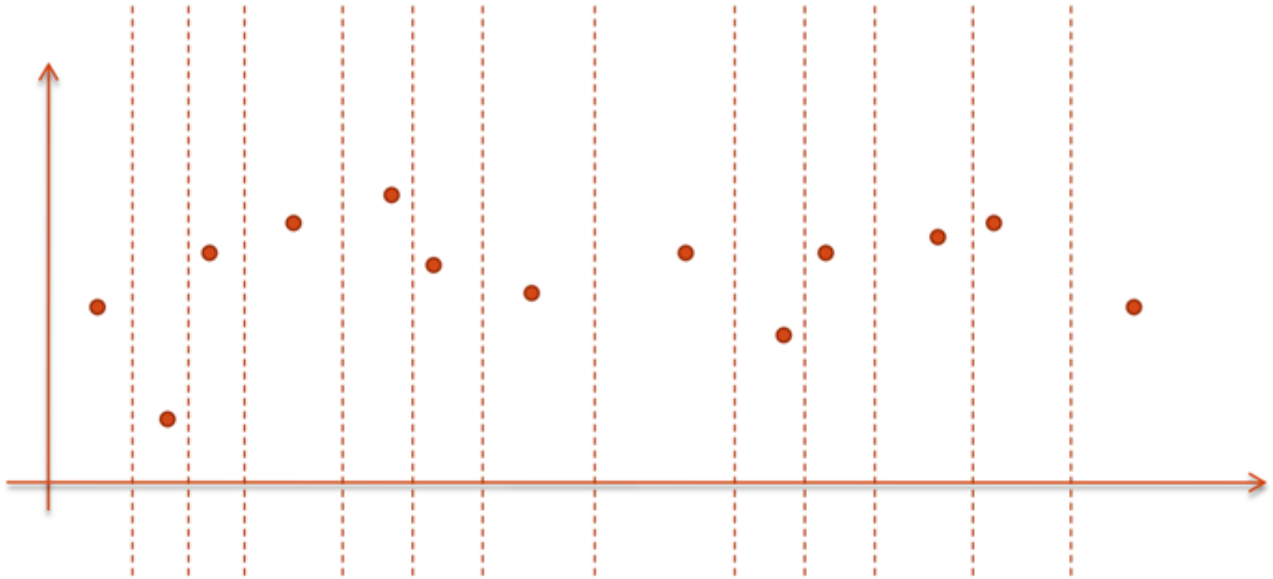
What is a Good Split?

- Prediction $\hat{y}$ is a constant in each interval.
- Residual sum of squares (RSS) in interval I: $RSS_I = \sum_{i \in I}(y_i - \hat{y}_i)^2$
- Minimal when $\hat{y} = \sum_{i \in I} y_i / n$



How Many Splits?

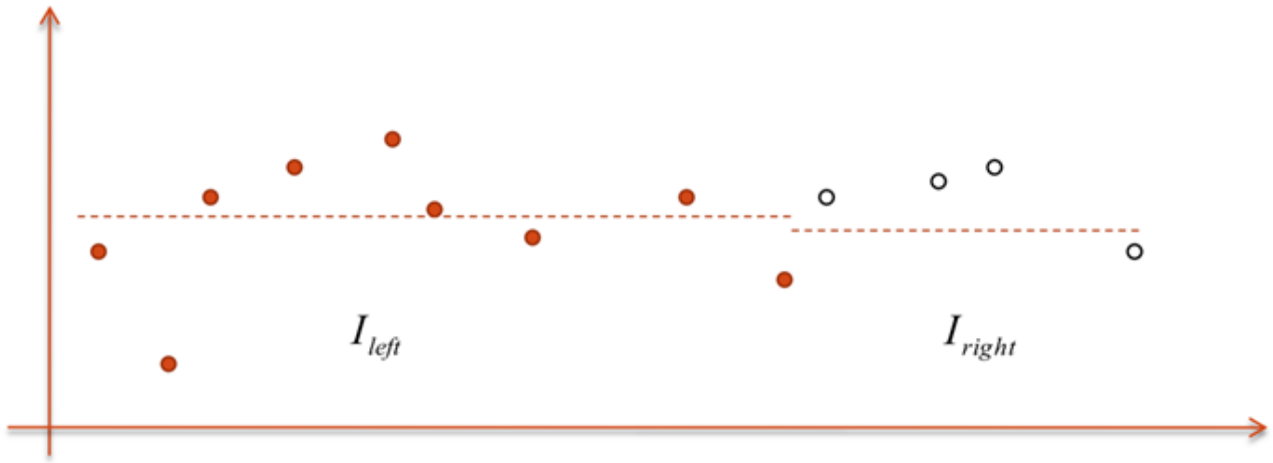
- Optimal constant function is the average of the outcome in the interval.
- Only important if observations are in the interval or not.
- Split somewhere in each gap between observations.



How Many Splits?

In an interval with n_I observations we have n_I possible splits

- Nothing fancy here, just try them all
- Chose the one with the lowest total RSS on the interval



$$RSS_I = RSS_{I_{left}} + RSS_{I_{right}}$$

Splitting Categorical Predictors

- Consider a two-category input (e.g. male-female)
  - Only one way to split, men versus women
  - Empty groups are not allowed
  - {male},{female} and {female},{male} is the same split
- Consider a three-category input {apple, orange, banana}
  - Three possible splits
    {apple} , {orange, banana}
    {apple, orange} , {banana}
    {apple, banana} , {orange}
- How many splits for a variable with k categories?
  - In today's exercises!

How Large Do We Grow the Tree?

- Stop splitting when a node contains too few observations
- For instance, do not split nodes with 10 or less observations

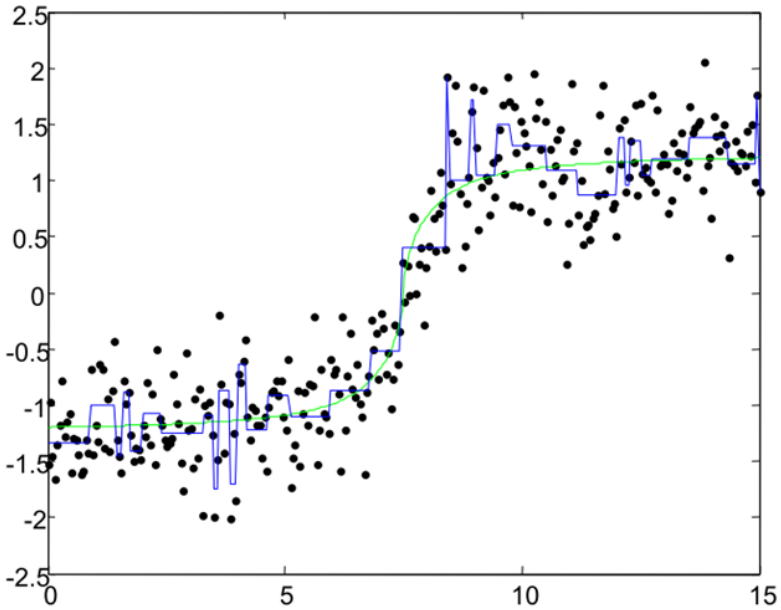
Tree-Growing Procedure

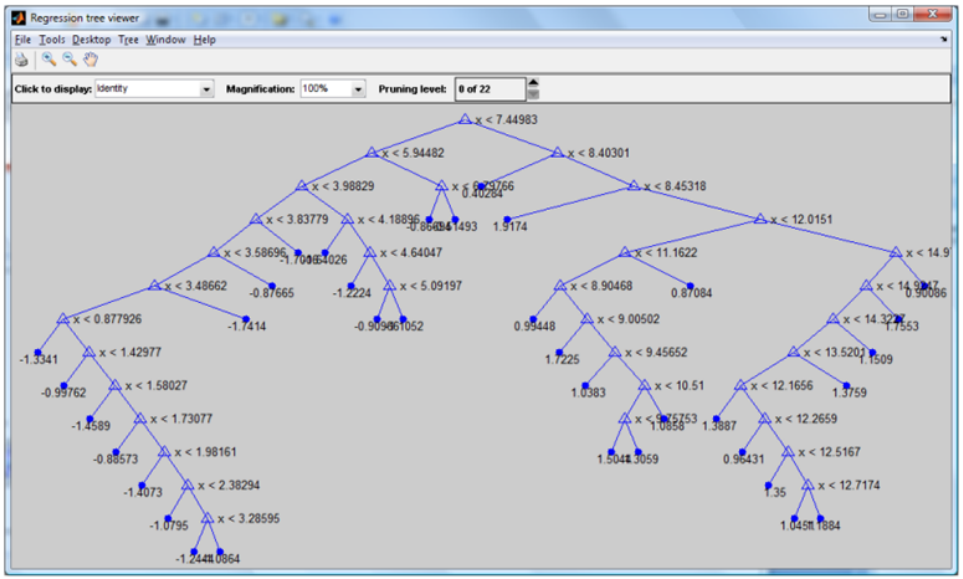
- First interval (node) is the entire range of X.
- For each variable
  - For each splitting position,
    calculate $RSS = RSS_{left} + RSS_{right}$
  - Remember position with the lowest RSS
- Split the variable with the lowest RSS at the corresponding position
- This produces a left and right sub-interval (child-nodes)
  - Split each of these into child nodes as above
  - Keep splitting nodes until a node contains too few observations
- Assign a constant function to terminal nodes, the average of the observations

Returning to the Example

`MinParent = 10`

(Matlab code for not splitting nodes with 10 or less observations)



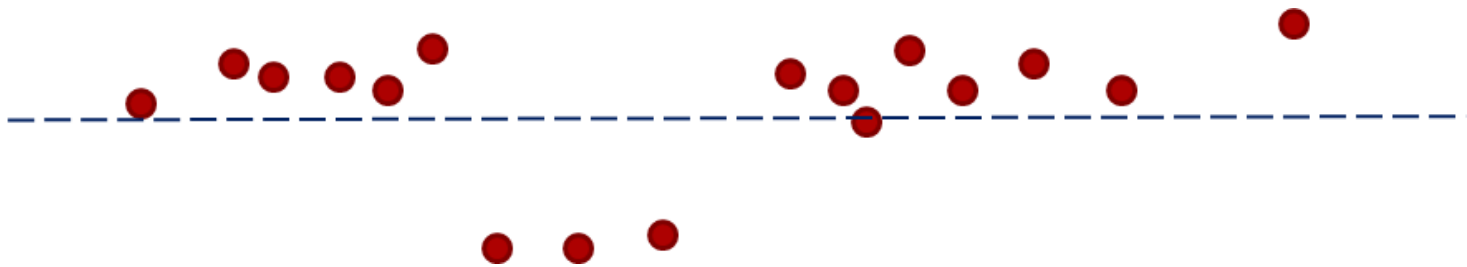


What do you think of this fit?

Finding the Right Sized Tree

- The tree was split too far - overfitting
- How do we know when to stop splitting?

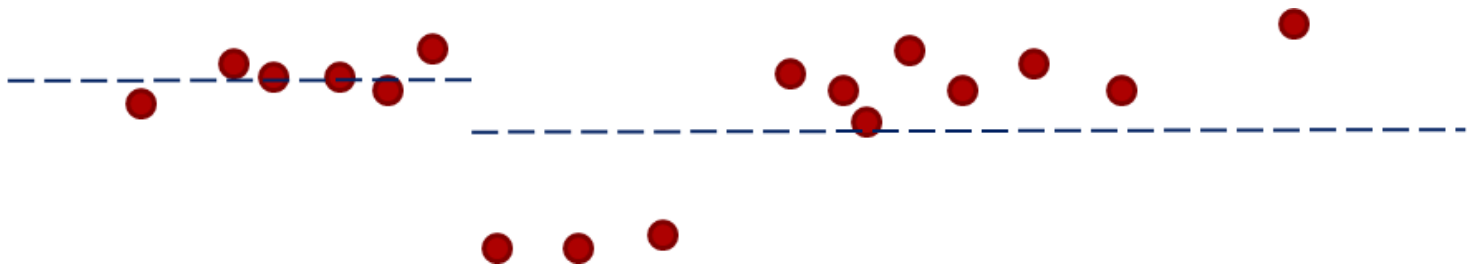
  Answer: we don't
  - A seemingly worthless split may lead to excellent splits below



Finding the Right Sized Tree

- The tree was split too far - overfitting
- How do we know when to stop splitting?

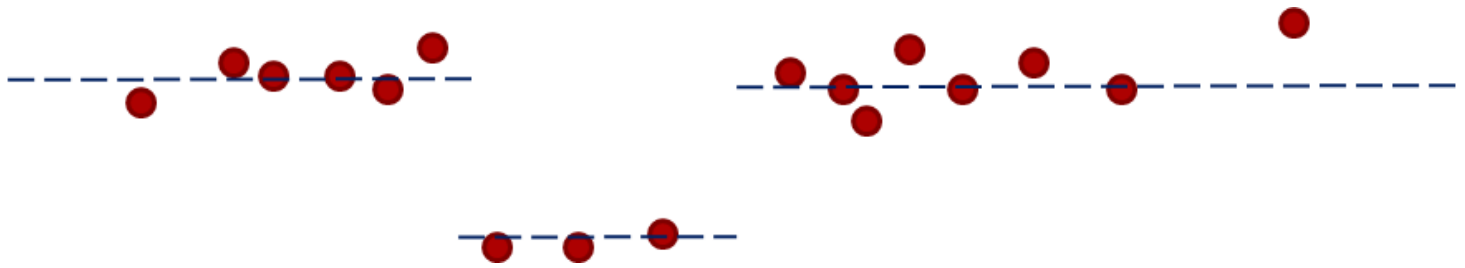
  Answer: we don't
  - A seemingly worthless split may lead to excellent splits below



Finding the Right Sized Tree

- The tree was split too far - overfitting
- How do we know when to stop splitting?

  Answer: we don't
  - A seemingly worthless split may lead to excellent splits below



Finding the Right Sized Tree

- We don't want to miss out on good splits
- Strategy: grow the tree really large (small `MinParent`)and then decide which splits were unnecessary and remove these.
- This is called **pruning** the tree
  - Pruning a node amounts to removing its sub-tree, thereby making a terminal node

Pruning Rule

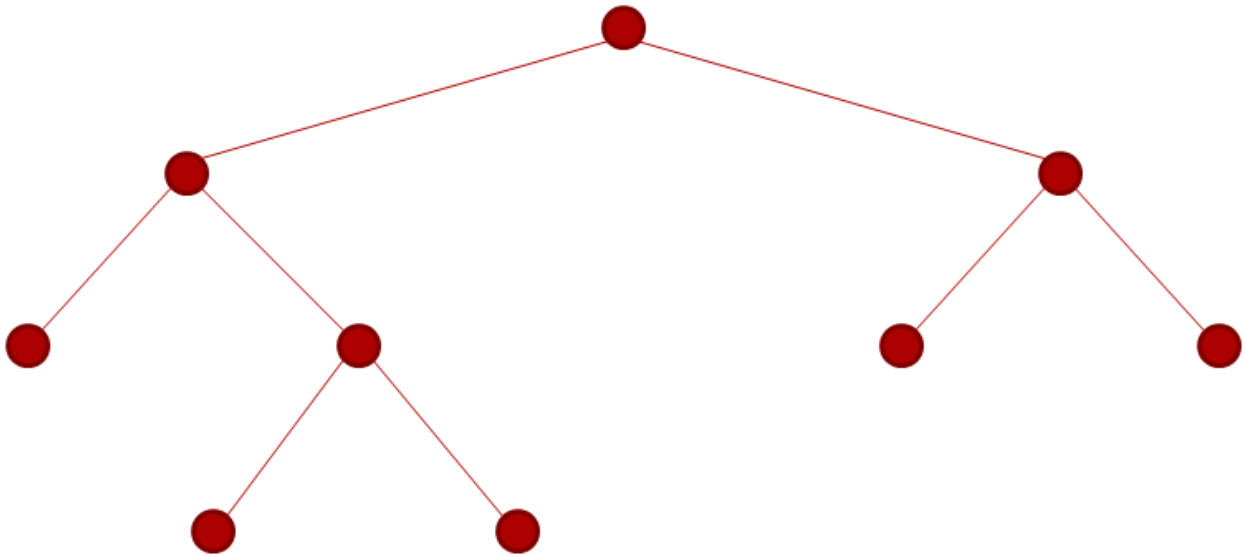
Prune the non-terminal node whose sub-tree gives the smallest **per node** reduction in RSS.

- Divide the reduction by the number of terminal nodes minus one

Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

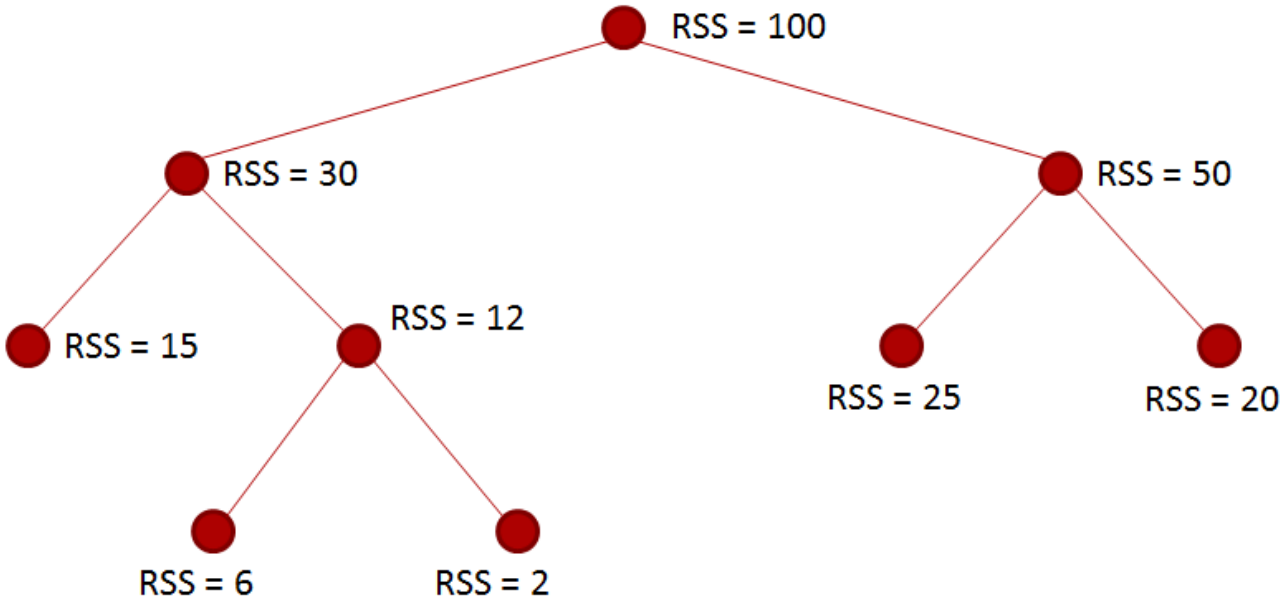
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

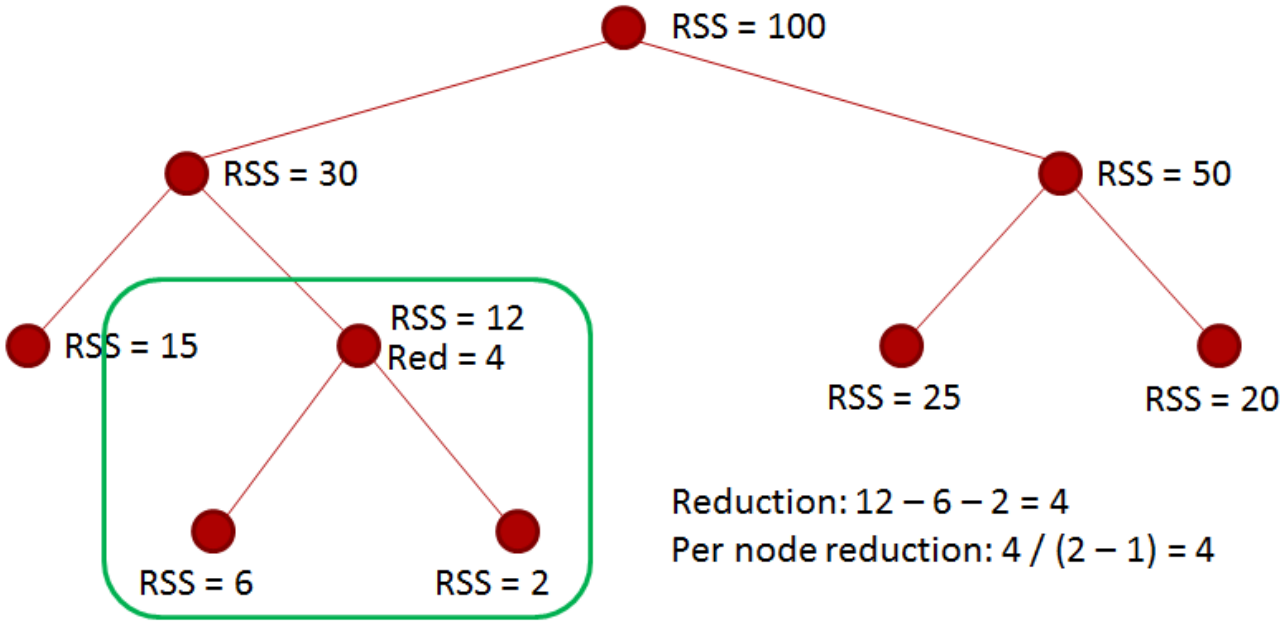
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

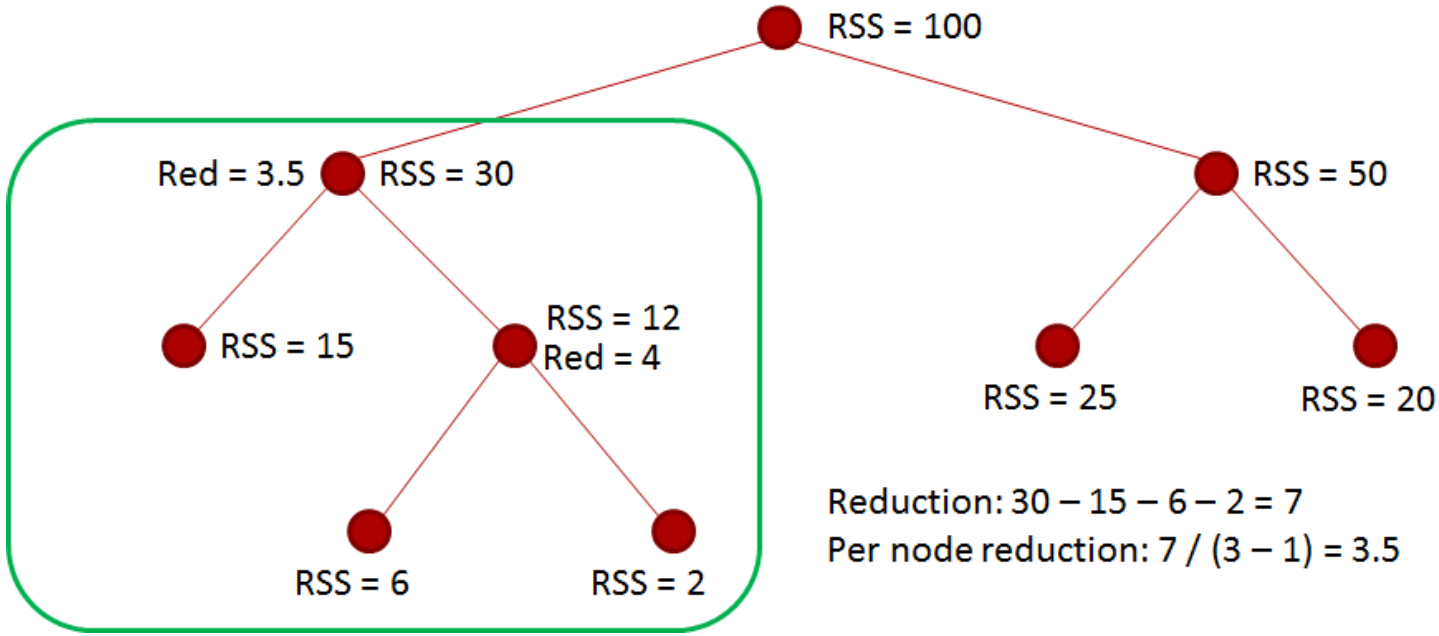
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

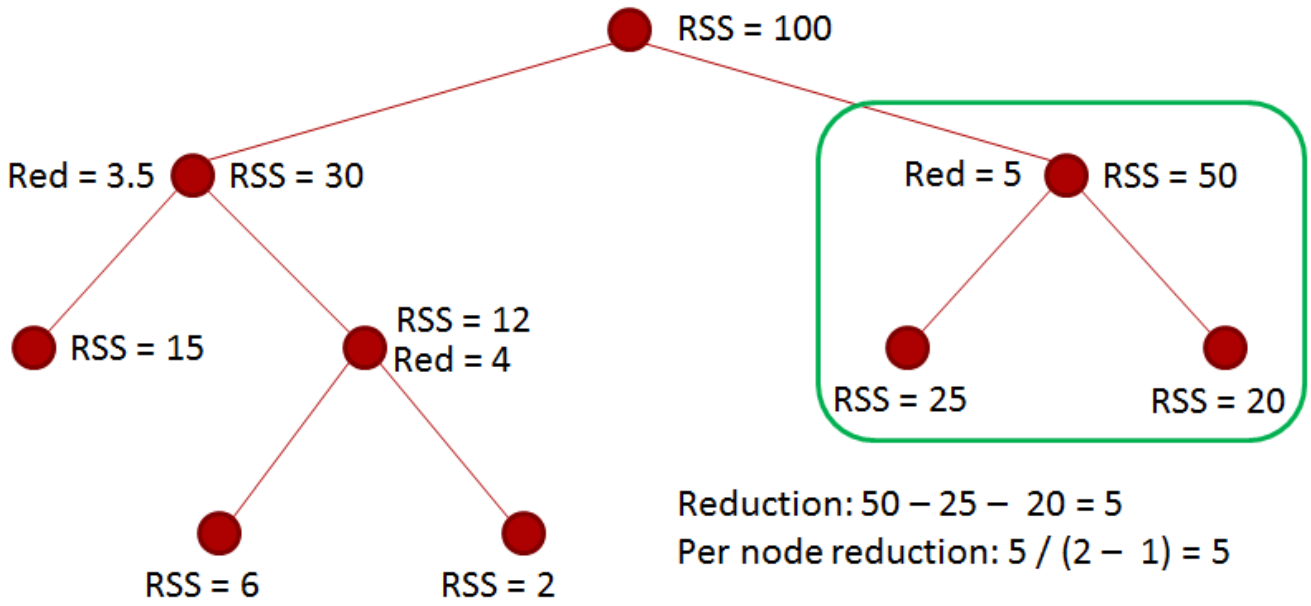
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

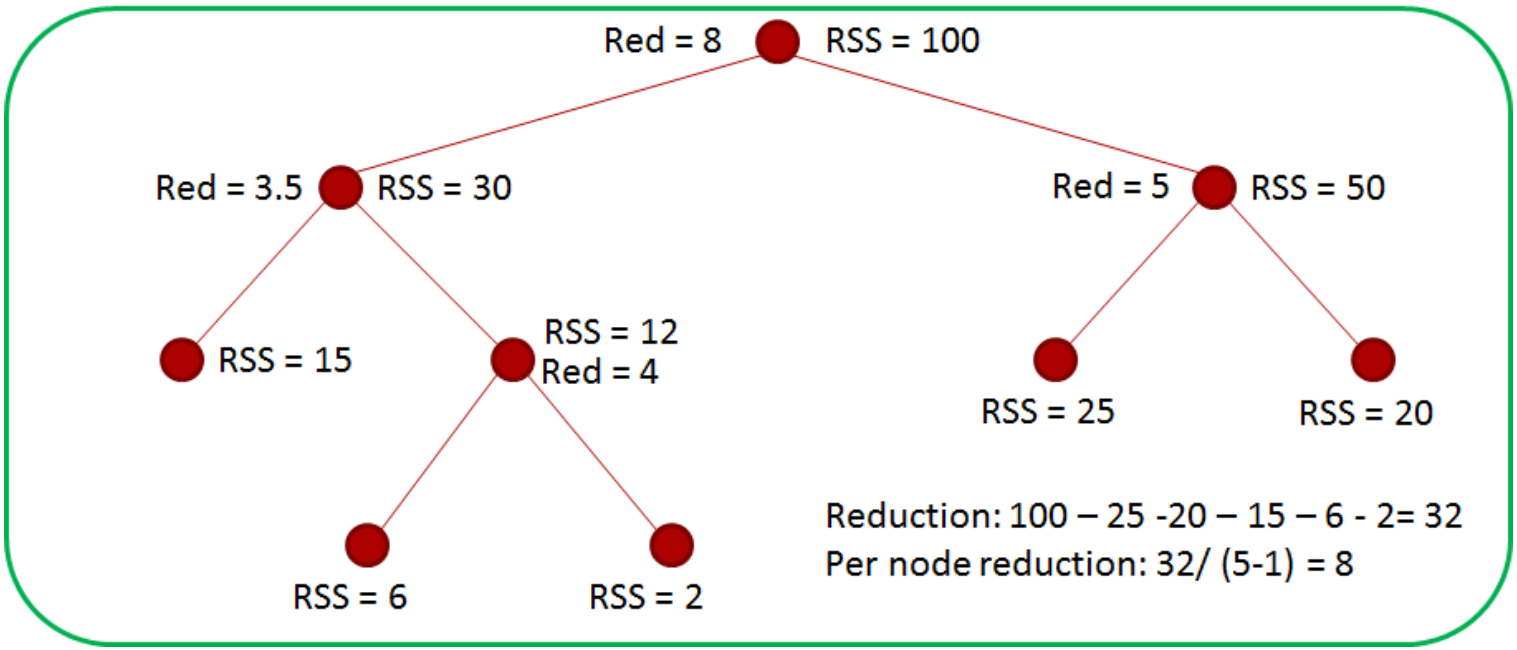
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

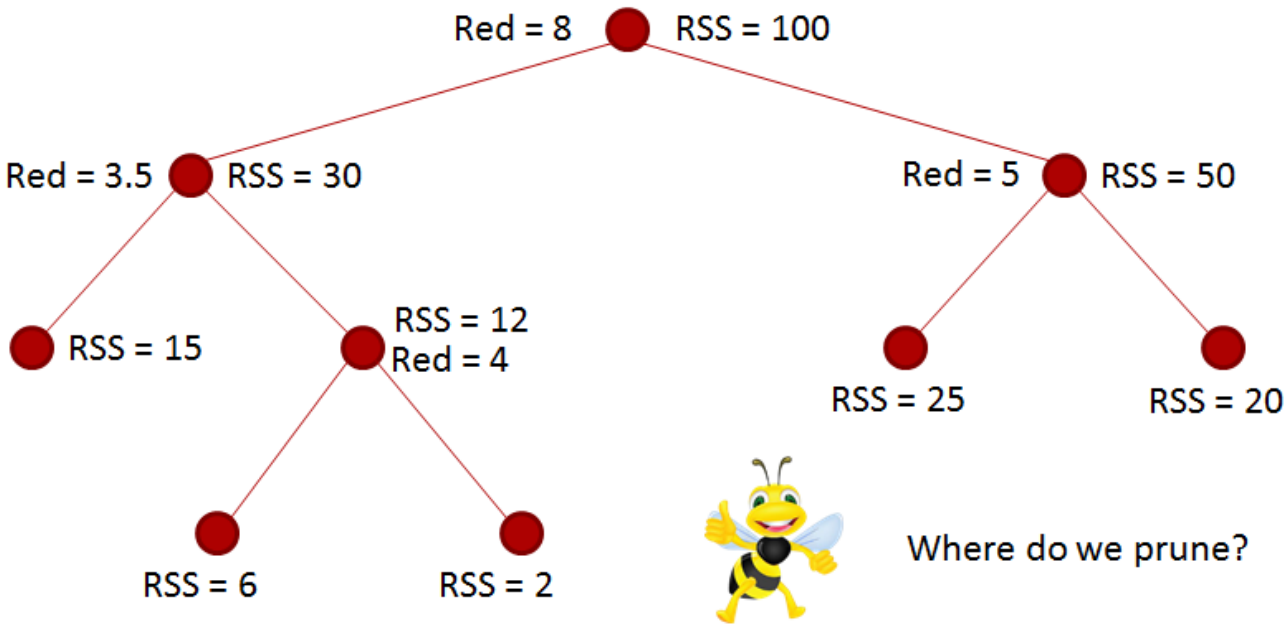
Example:



Pruning the Tree

Weakest-link pruning: prune branches that contribute the least to lowering RSS

Example:

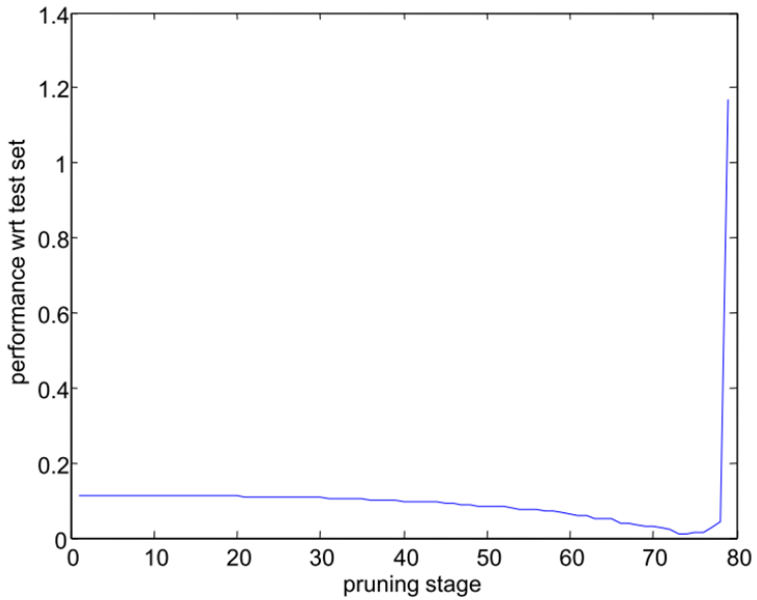


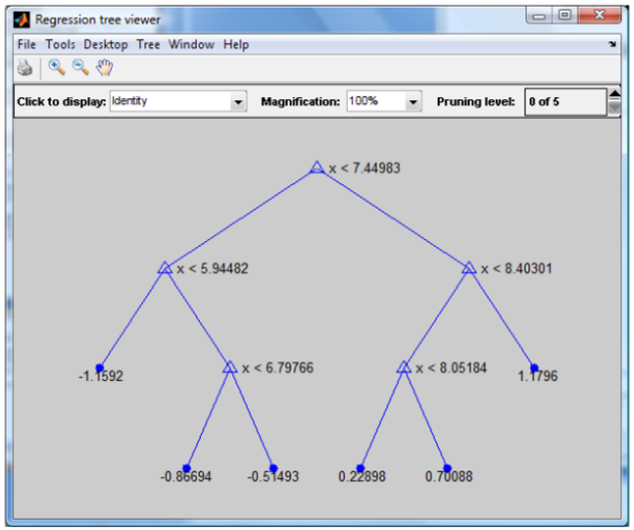
When to Stop Pruning?

- If we keep pruning, we will end up with just the root node
- Which of all these sub-trees do we choose?
- Two approaches
  - Independent test set
  - Cross-validation

Using an Independent Test Set

- As we prune our way towards the root node, evaluate the performance of each sub-tree with respect to the test set.
- Continue all the way to the root node
  - Choose sub-tree with best performance





Using Cross-Validation

Three alternatives...

- Cross validate **all possible trees**, does not work in practice - too many trees.
- Use a **tuning parameter**. Choose the tree that minimizes,

$$RSS(T) + \alpha|T|$$

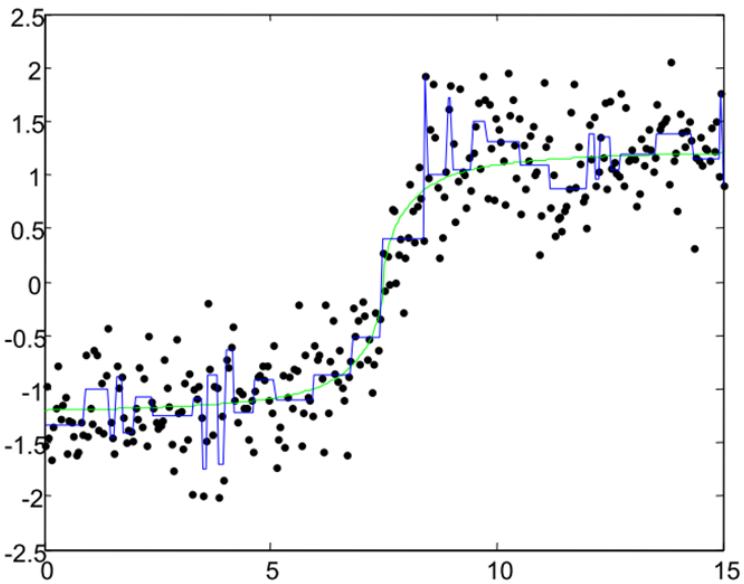
$$|T| = \text{ \# end-nodes}$$

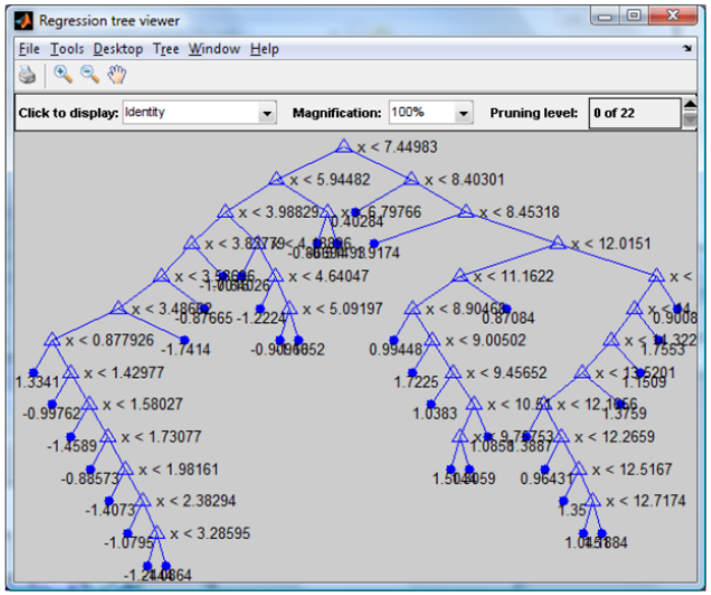
  Find the tuning parameter α using cross validation

- Use the cross validation to determine the optimal value of the minimum number of **observations in a terminal node**. Matlab parameter '`MinLeaf`'. Then use the full grown tree without pruning.

Regression Example Revisited

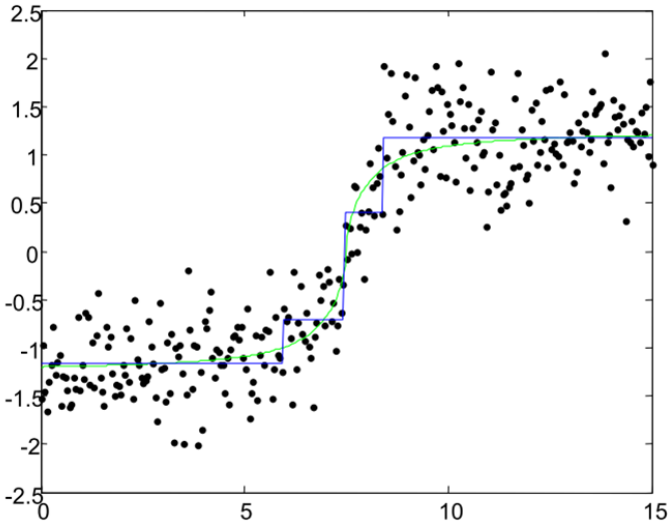
`MinParent = 20` full tree

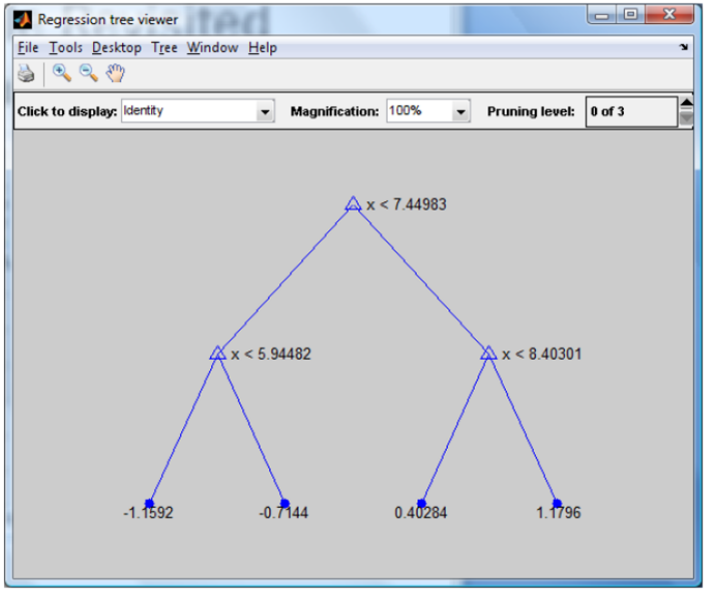




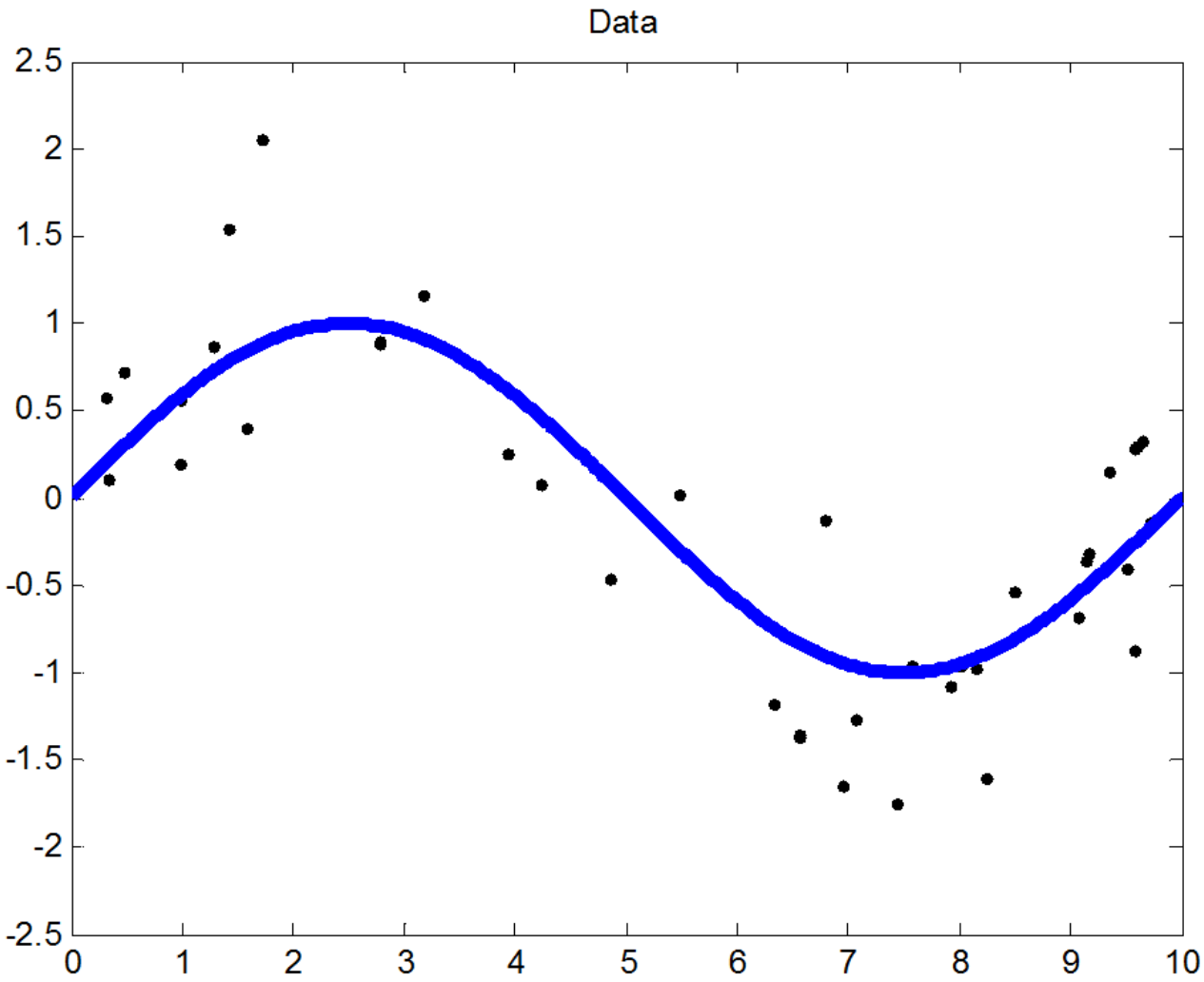
Regression Example Revisited

Best sub-tree chosen by 10-fold cross-validation.

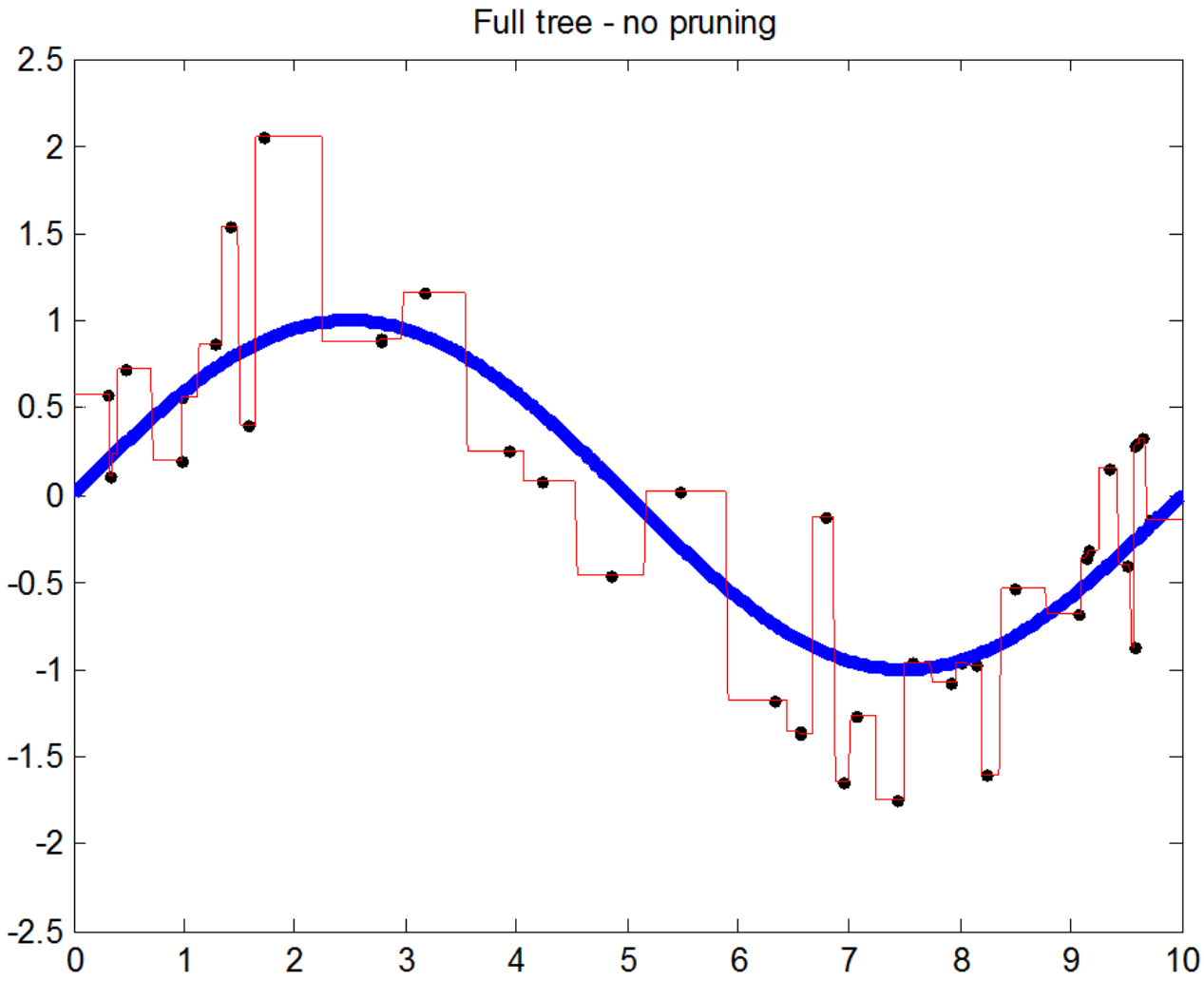




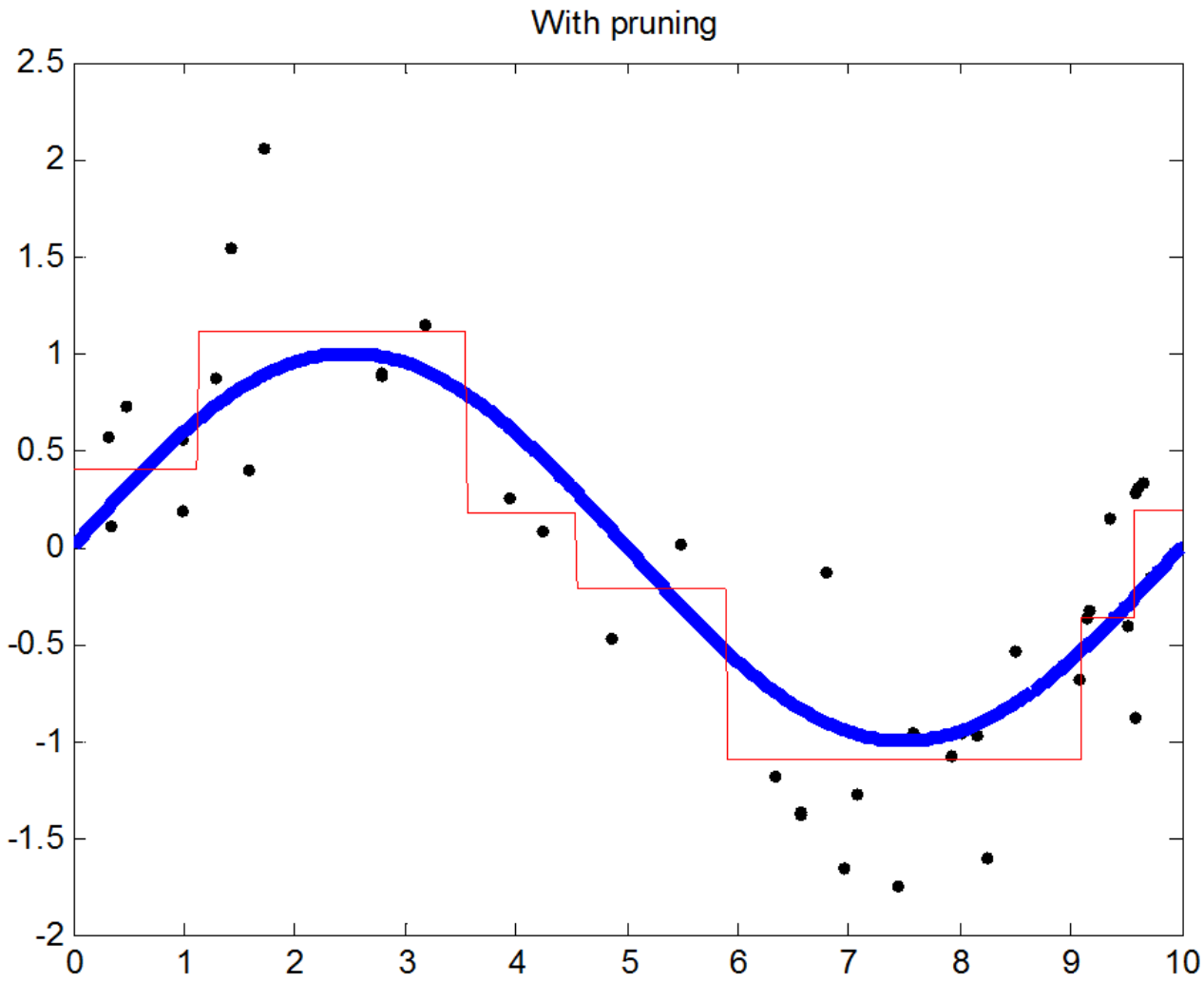
Bias and Variance Trade-Off



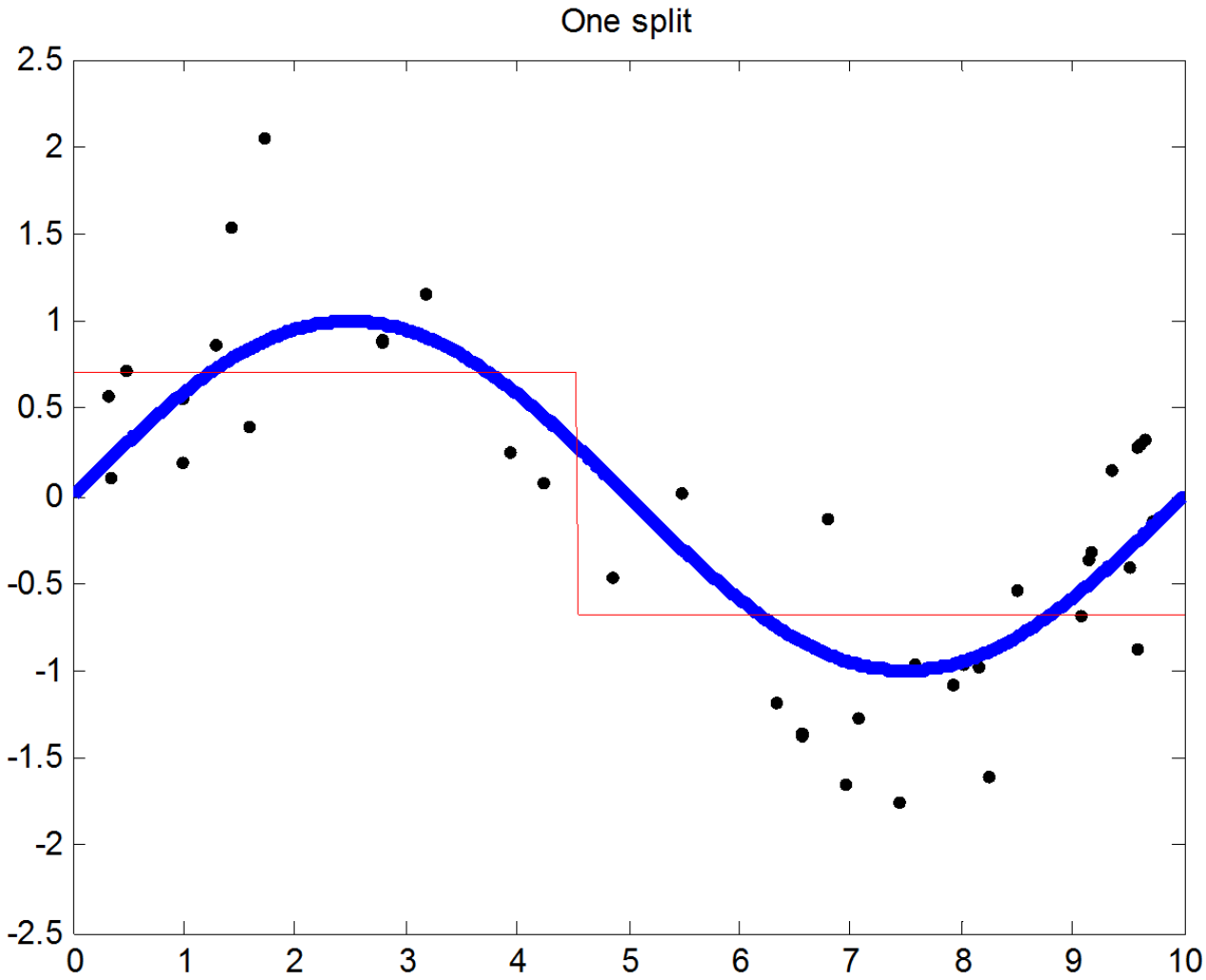
Bias and Variance Trade-Off



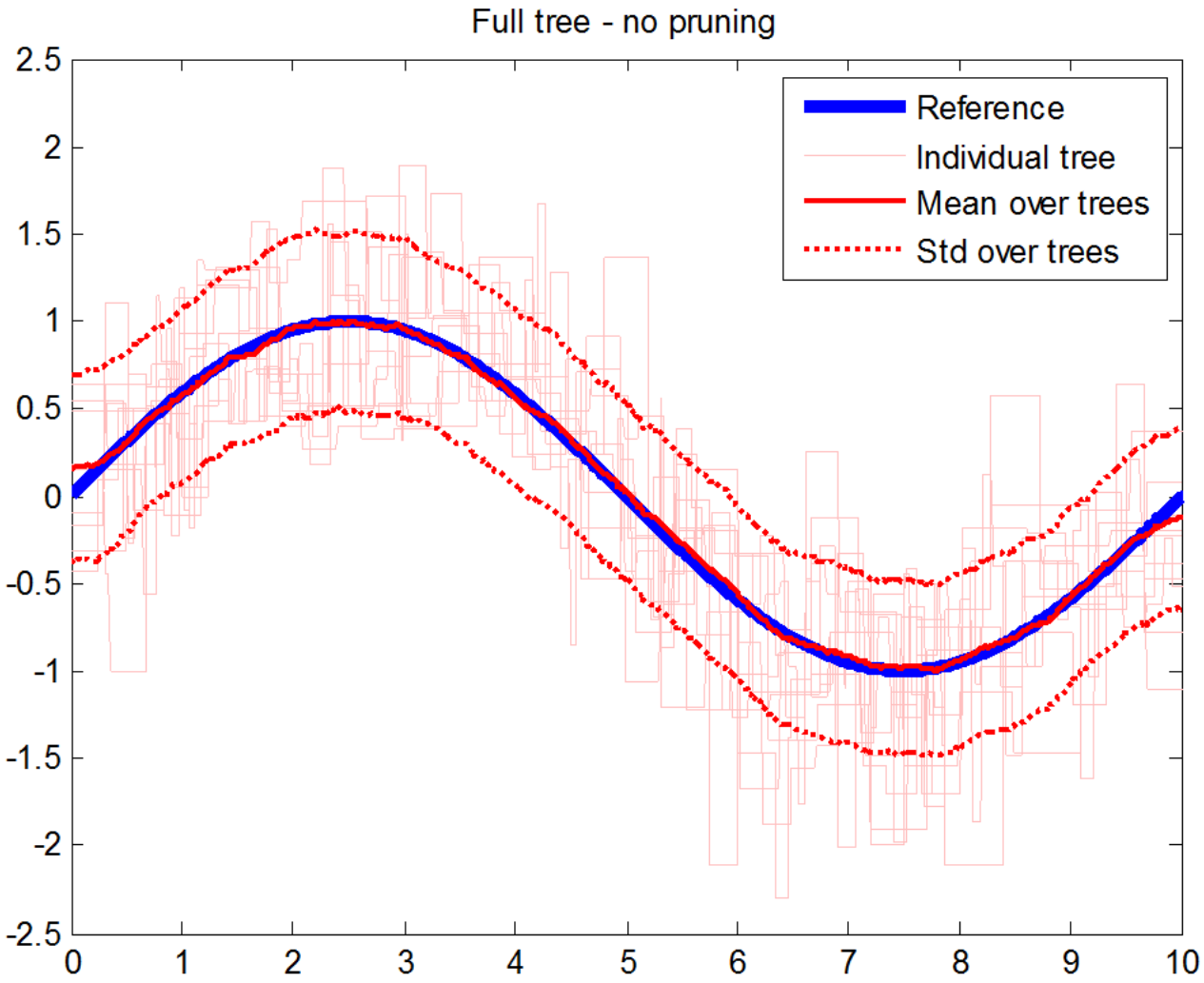
Bias and Variance Trade-Off



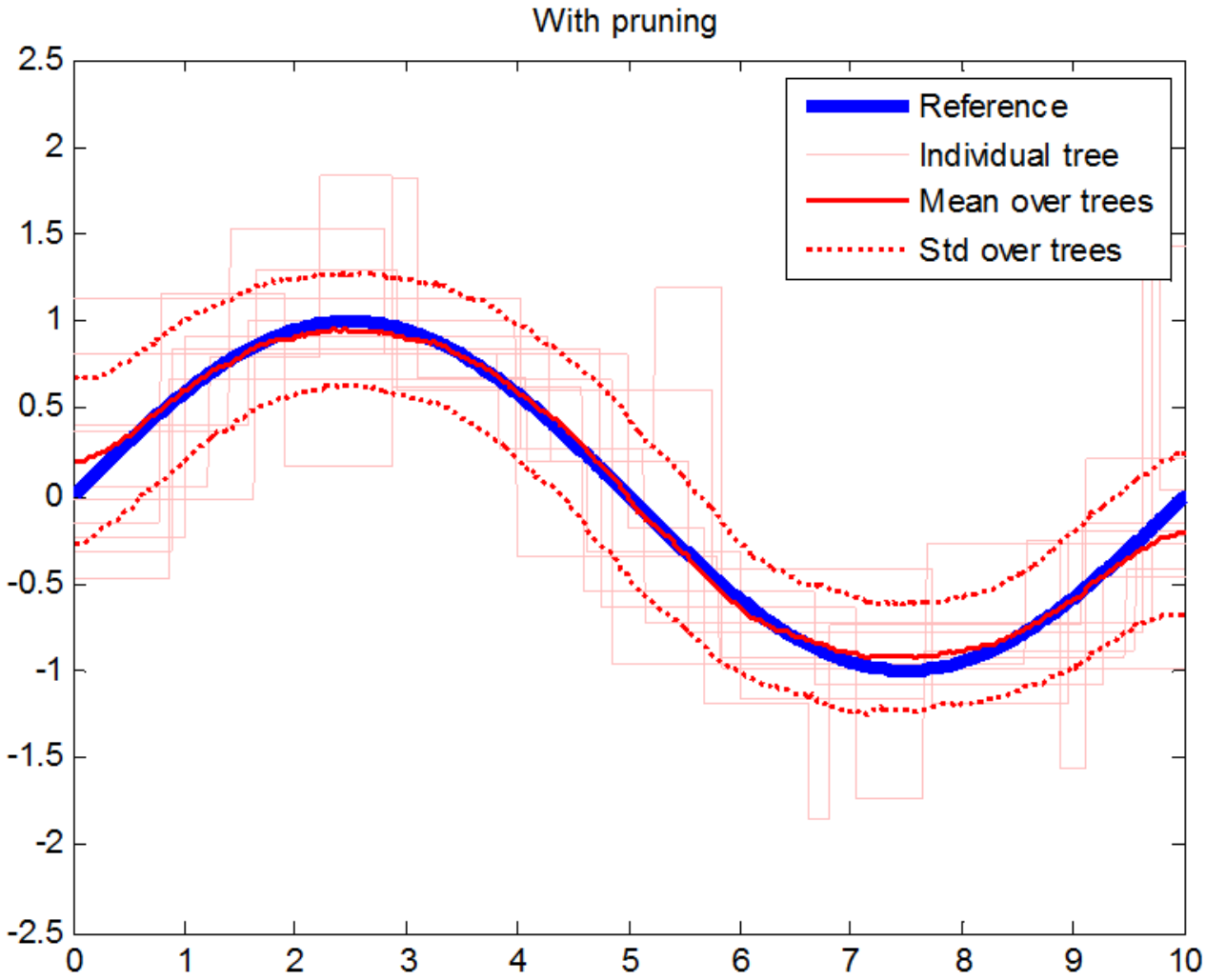
Bias and Variance Trade-Off



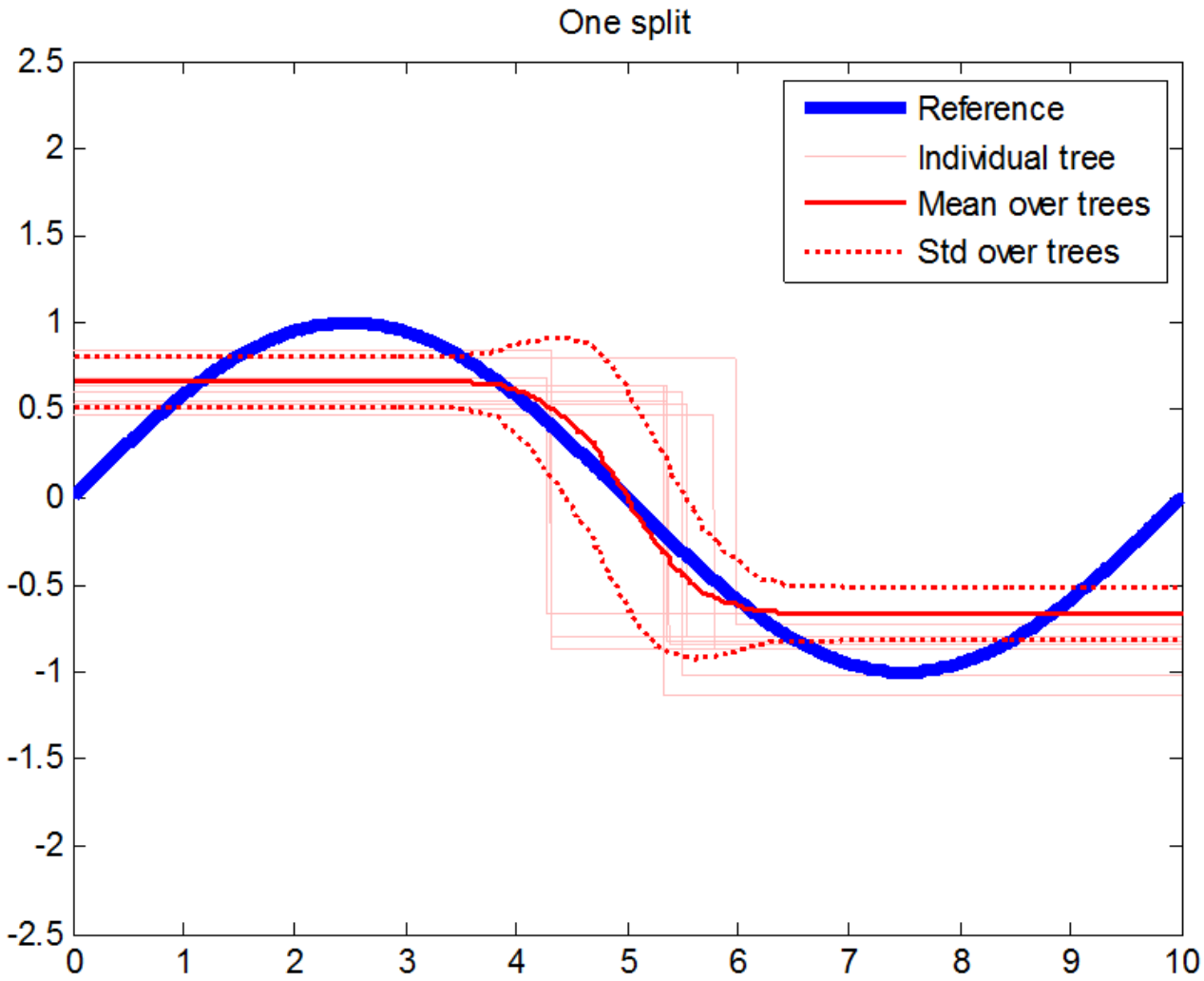
Bias and Variance Trade-Off



Bias and Variance Trade-Off



Bias and Variance Trade-Off



Bias and Variance Trade-Off

What do we conclude about regression trees in terms of

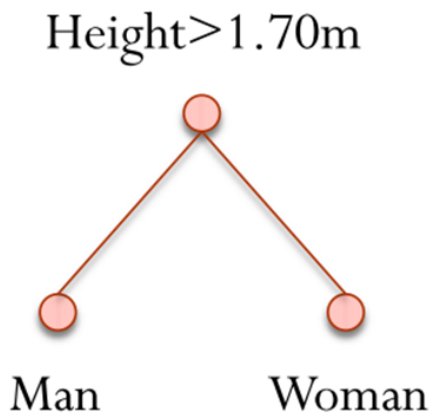
- Bias
- Variance

Classification Trees

- Classification Method

Classification Trees

Terminal node assigns a class instead of a constant



Only difference: criteria for splitting nodes and pruning the tree

Node Values

With **regression trees** we transform a new observation into a constant.

- The constant was derived from the training data.
- It was the **mean** of the output variable of the training observations in the node.

For **classification trees** we assign new observations to a certain class

- The class is derived from training data
- It is the **majority class** of the training observations in the node

Model Error

Regression trees

- For regression trees we used RSS as a measure of node impurity

Classification trees

- For classification trees we have a few options
  - Missclassification rate
  - Gini index
  - Cross-entropy
- They all favor the split that increases purity the most.
  - Typically the predictive performance is not that different
  - The shape of the trees might, however, be very different

Node Impurity for Classification Trees

In a specific node, representing a region R with N observations, let

$$\hat{p}_k = \frac{1}{N} \sum_{x_i \in R} \mathbb{1}(y_i = k)$$

Classify observations in the node to class,

$$K = \arg\max_k \hat{p}_k$$

Measures of impurity within a node,

Misclassification error: $Q = \frac{1}{N} \sum_{i \in R} \mathbb{1}(y_i \neq K) = 1 - \hat{p}_K$

Gini index: $Q = \sum_{k \neq k'} \hat{p}_k \hat{p}_{k'} = \sum_k \hat{p}_k (1 - \hat{p}_k)$

Cross-entropy (deviance): $Q = -\sum_k \hat{p}_k \log \hat{p}_k$

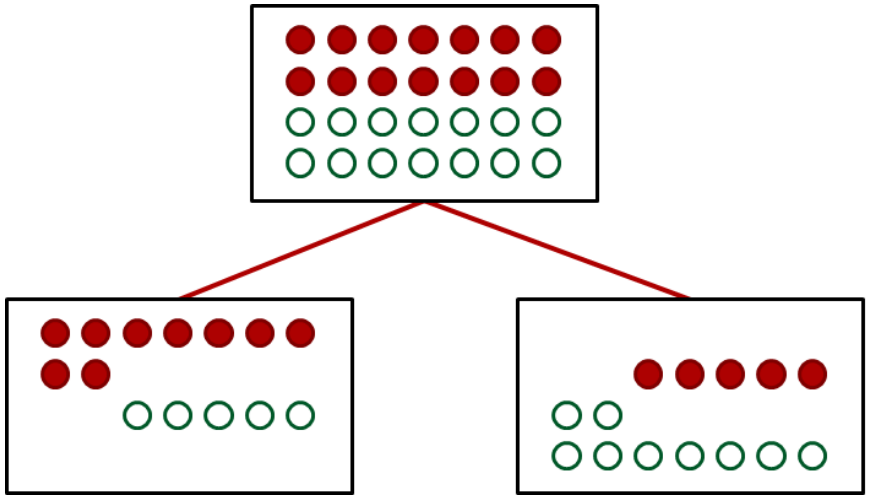
From Node Impurity to Split Criterion

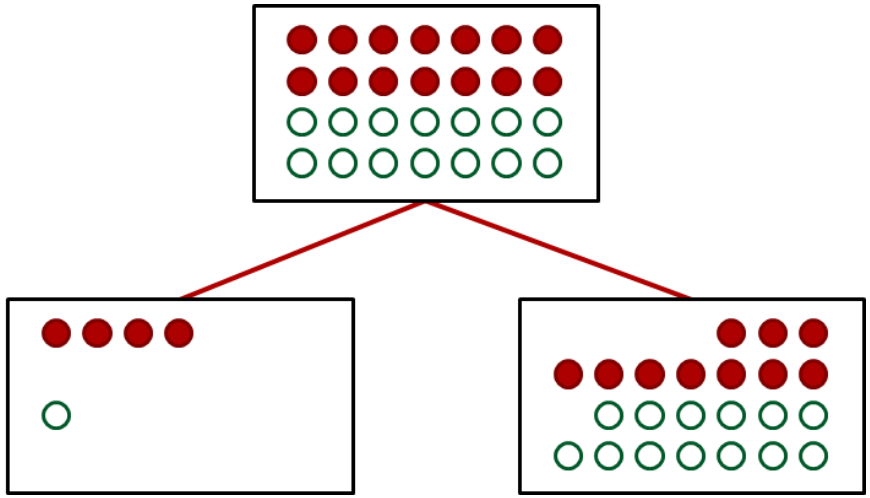
- The node impurity is weighted with the number of observations in each node.
- The split decision is based on the split that **minimizes**,

$$N_{left} Q_{left} + N_{right} Q_{right}$$

N, the number of observations in left and right node
Q, node impurity for left and right node

Comparing Splits



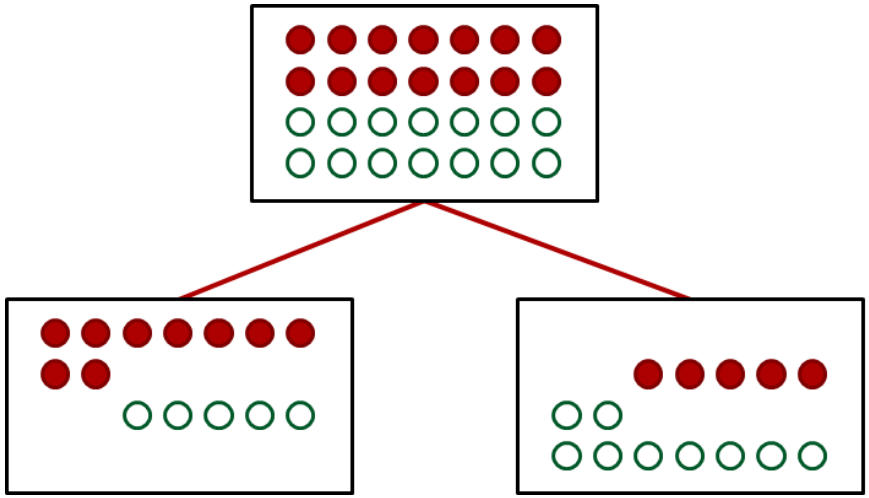


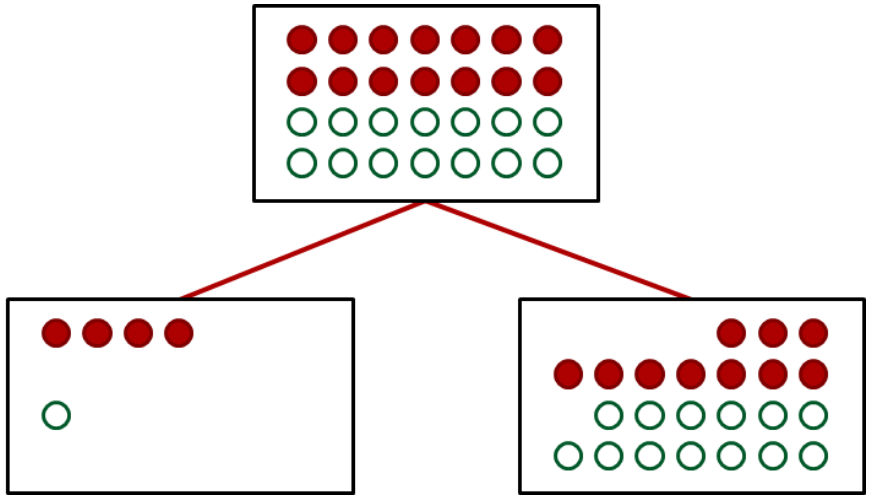
Missclassification error
Left node: 5/14
Right node: 5/14
Split criterion
14(5/14) + 14(5/14) = 10

Missclassification error
Left node: 1/5
Right node: 10/23
Split criterion
5(1/5) + 23(10/23) = 11

Lower is better!

Comparing Splits



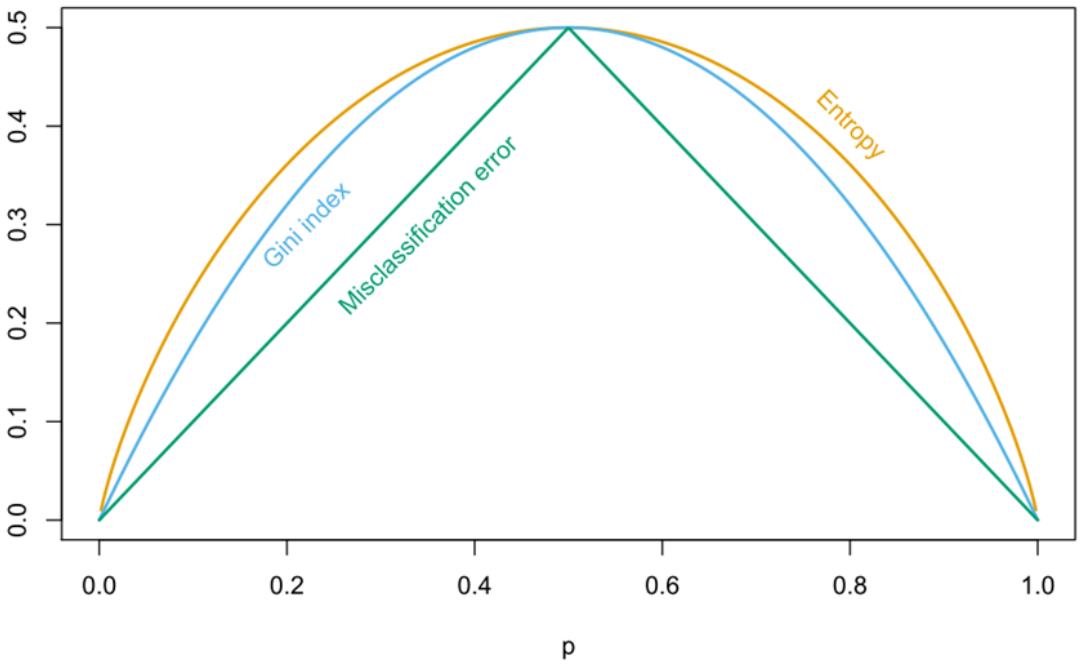


Gini index
Left node: ...
Right node: ...
Split criterion
...

Gini index
Left node: ...
Right node: ...
Split criterion
...

Node Impurity for Classification Trees

- Node impurity measures for a two-class problem.
- X-axis: proportion of samples belonging to class 2.
- Entropy and Gini index are better measures for growing tree because they are more sensitive to node probabilities.



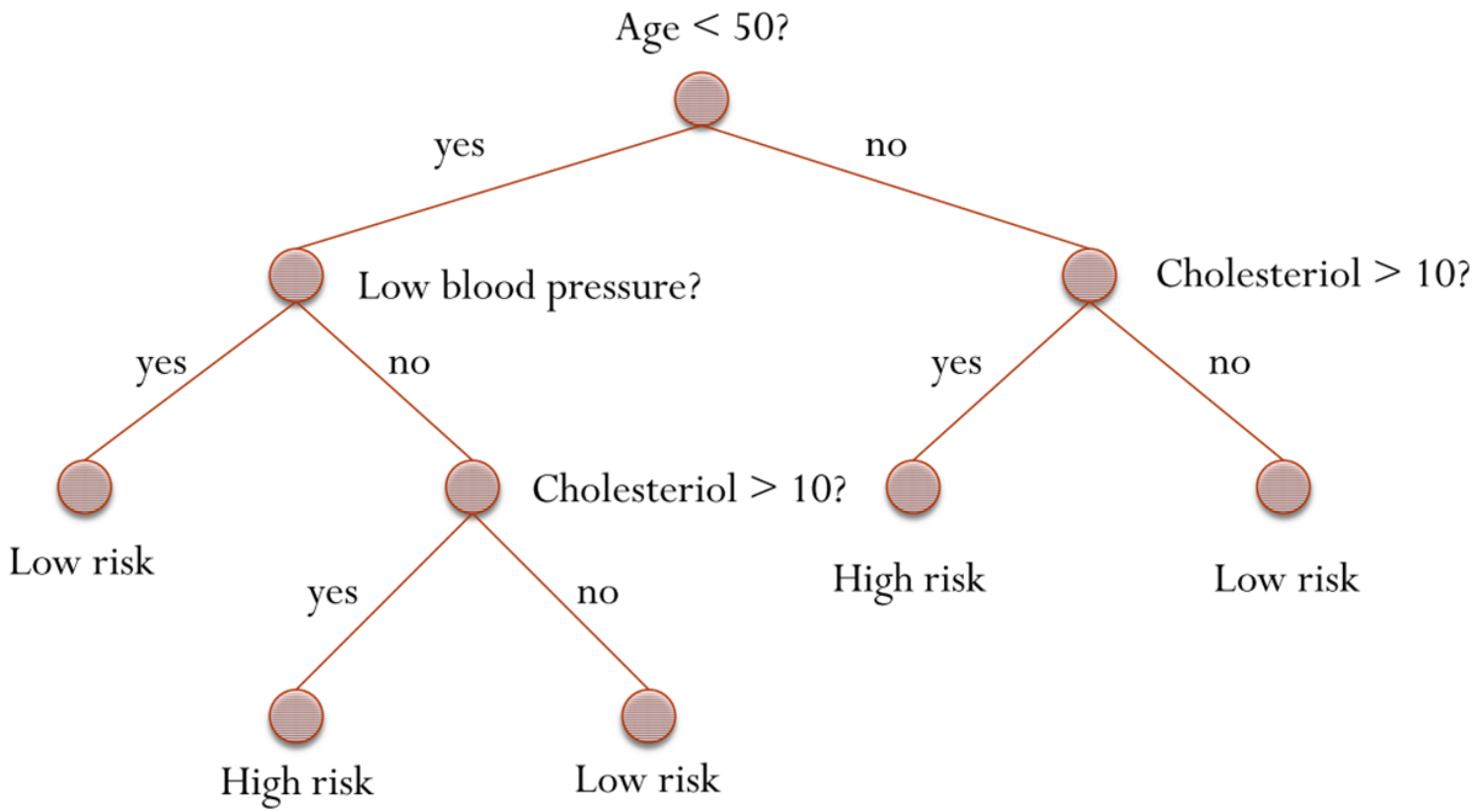
Standard Practice

- Use **Gini index** as split criterion when **building** the tree.
- Use **missclassification rate** as criterion when deciding which node to **prune**.

Benefits of a Tree Structure

Interpretability!

Consider the following (classification) tree on heart diseases



Interpretability

- CART is popular in medical sciences because it may represent the way doctors reason.
- A single tree describes the entire partitioning on the input space.
- With $p > 3$ input variables, the partition (cf. knot positions) are difficult to visualize.
  - But a tree representation is always possible.
- A large tree might be difficult to interpret anyway...

Missing Data

Incomplete data are common in many applications.

We can always

- Delete the observation
- Replace with mean or median

For trees we can

- Introduce and **extra category** “missing” - if it is a categorical variable.
- Use a **surrogate variable**. In each branch, have a list of alternative variables and split points - as a backup.
  - Matlab does this.

Bagging

- Ensemble method
- Many models on bootstrap samples
- Output from all models aggregated into one model

Review: Bootstrapping

Start with data

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \\ 5 & 10 \end{bmatrix}$$

sample with replacement

$$X_1' = \begin{bmatrix} 3 & 6 \\ 2 & 4 \\ 3 & 6 \\ 5 & 10 \\ 3 & 6 \end{bmatrix}, X_2' = \begin{bmatrix} 4 & 8 \\ 4 & 8 \\ 2 & 4 \\ 1 & 2 \\ 3 & 6 \end{bmatrix}, X_3' = \begin{bmatrix} 1 & 2 \\ 5 & 10 \\ 2 & 4 \\ 3 & 6 \\ 4 & 8 \end{bmatrix}$$

Bagging

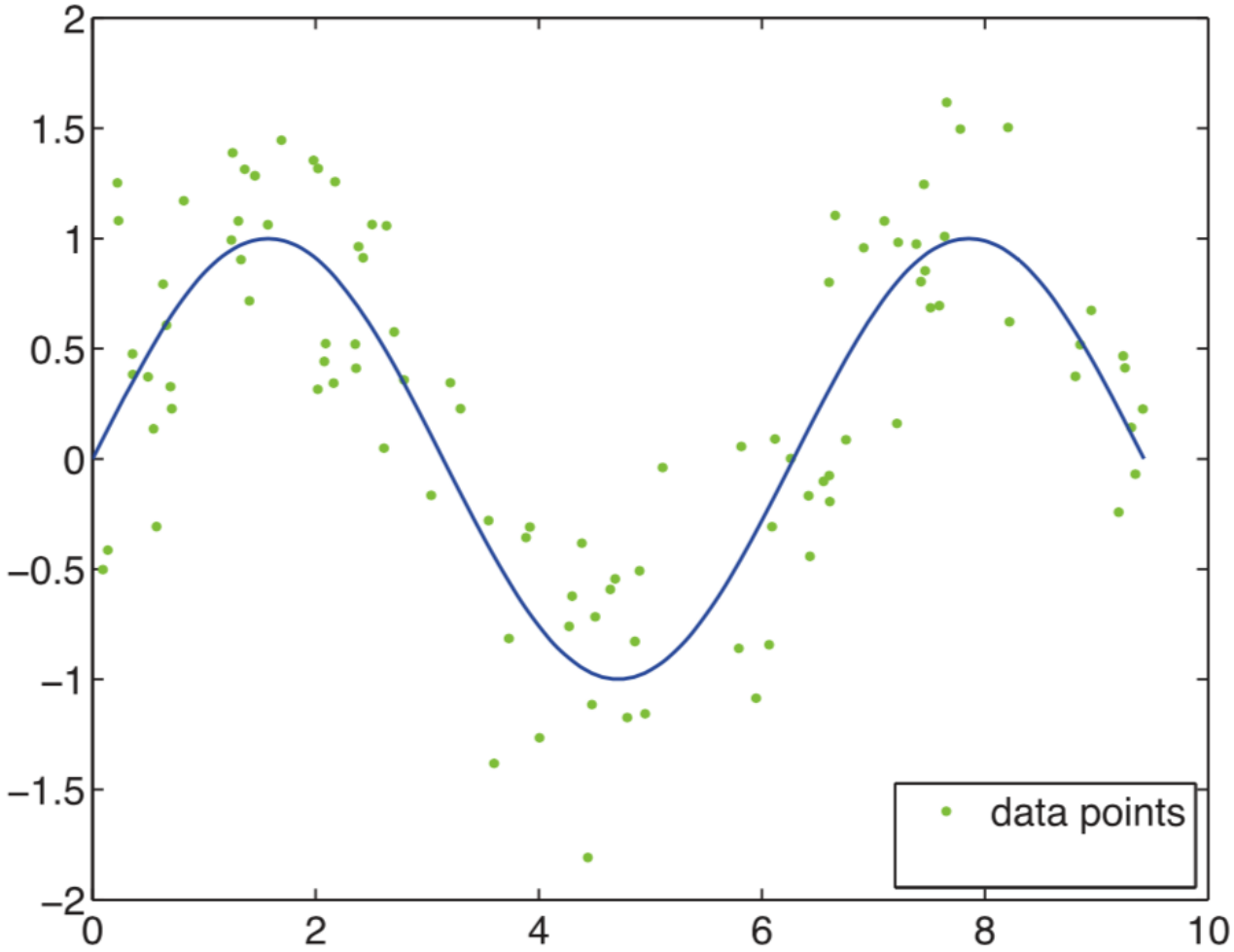
- Use bootstrap to improve prediction (not to tune parameters or assess prediction errors)
- Bagging averages predictions over a collection of bootstrap samples
- Average many noisy but approximately unbiased models and thereby reduce the variance

Bagging Algorithm

1. Make B boostrap samples of of size N
2. For $b = 1$ to B repeat
   1. Fit a model using the bth sample and make the prediction $\hat{y}_b$
3. The bagging estimate is given by the average of the B predictions

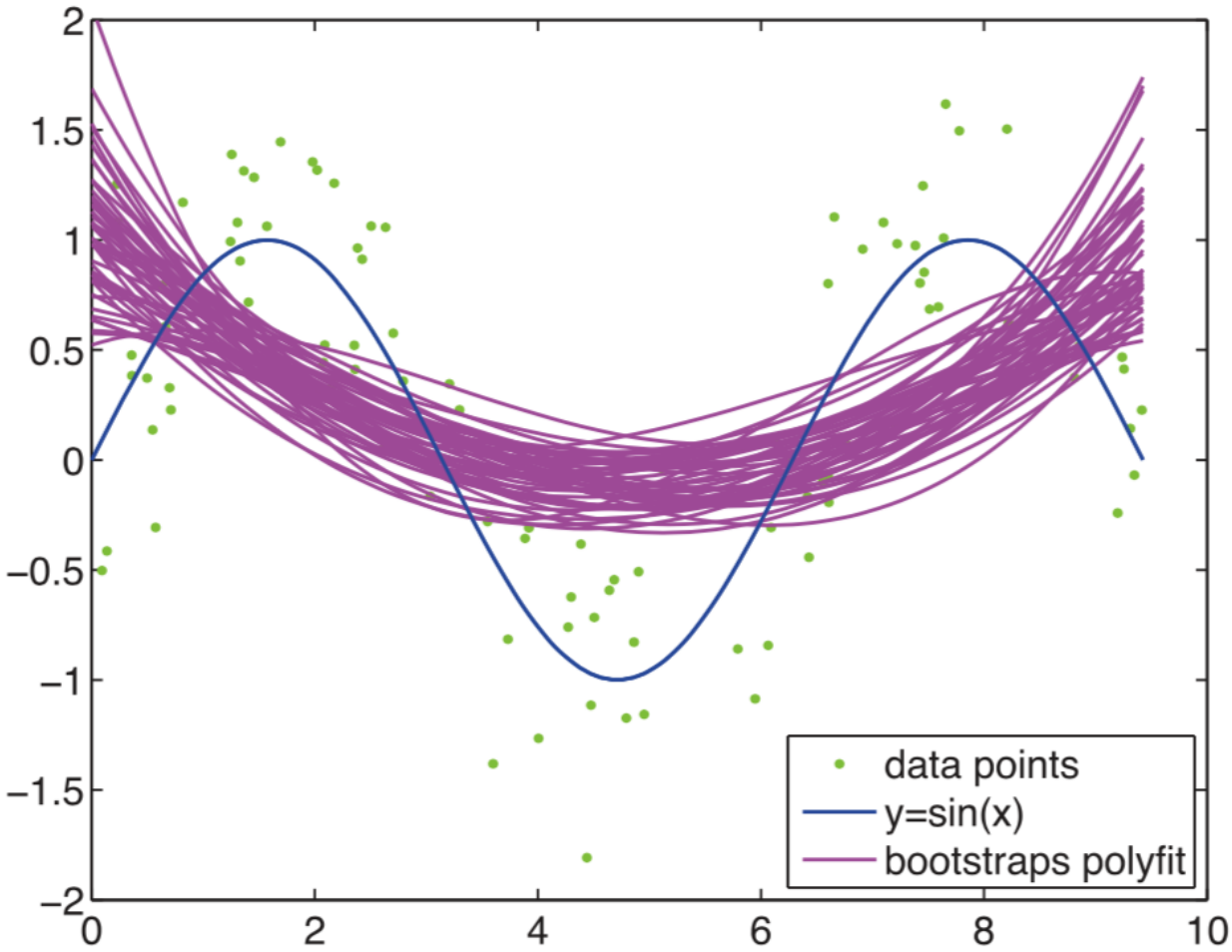
$$\hat{y}_{\text{bagging}} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b$$

Bagging Example - sin(x)



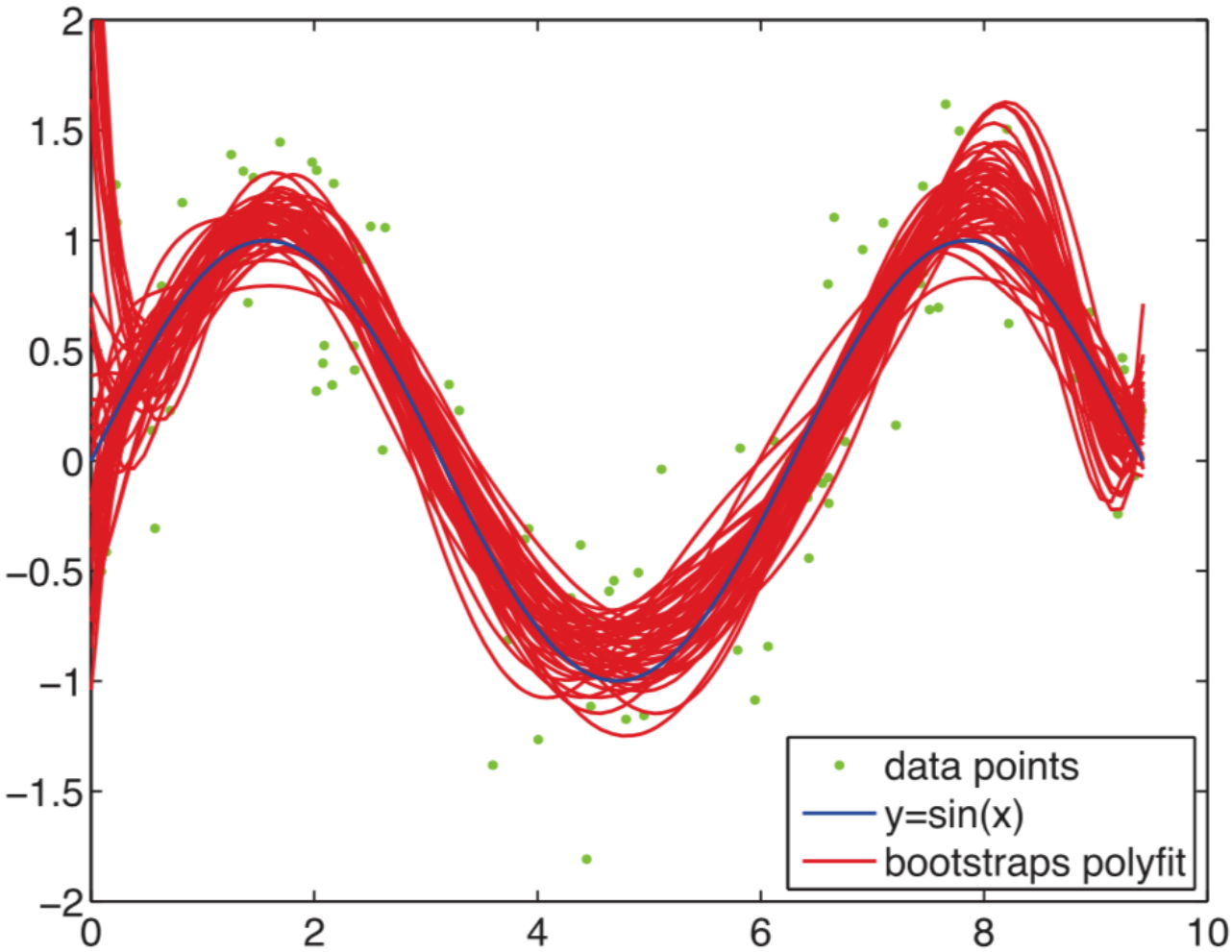
100 data points with white noise added from $y = \sin(x)$.

Bagging Example - sin(x)



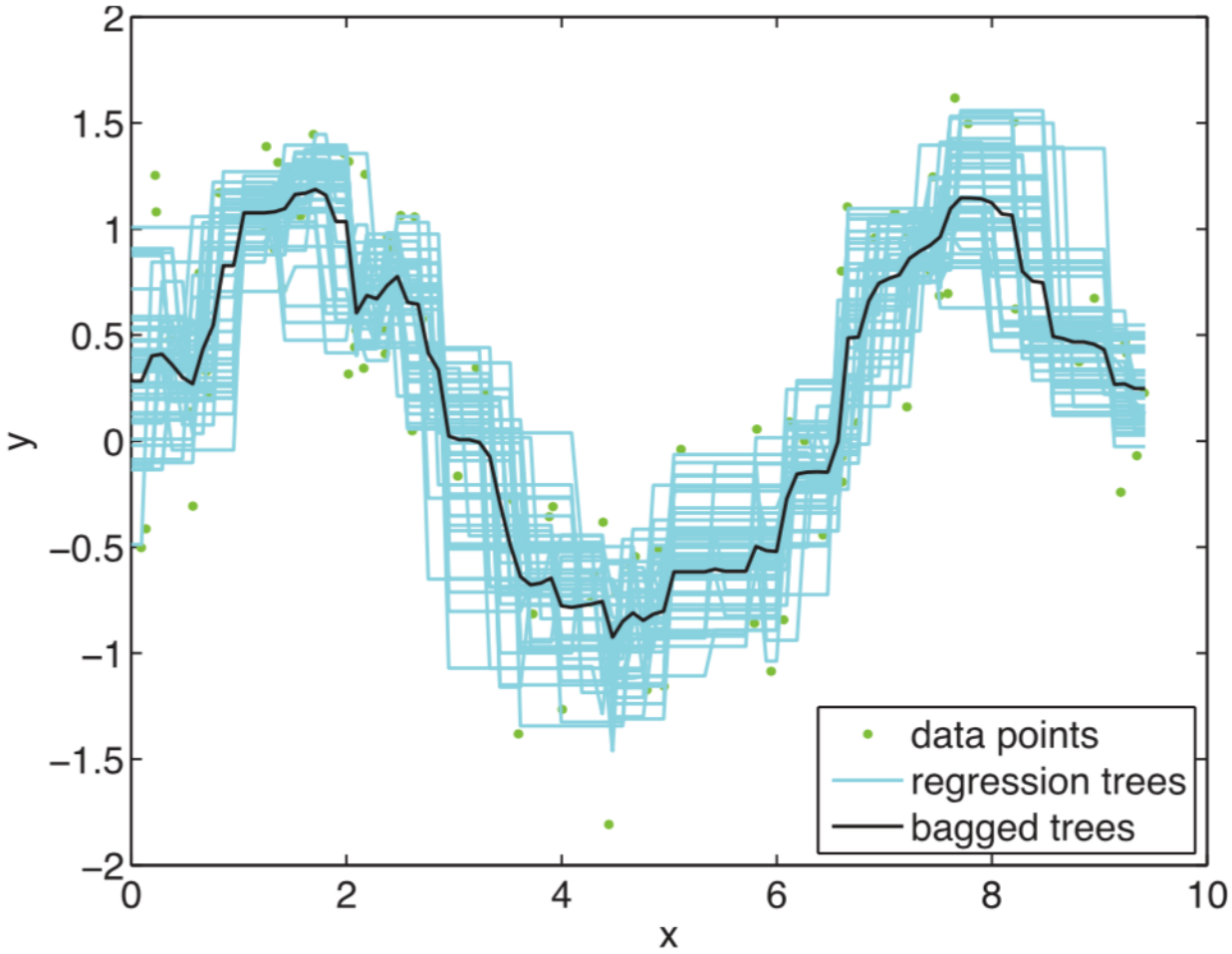
Fits of a 3rd degree polynomial to 50 bootstraps of data.

Bagging Example - sin(x)



Fits of a 9th degree polynomial to 50 bootstraps of data.

Bagging Example - sin(x)



Fits of regression trees to 50 bootstrap samples and the bagged tree

Bagging - Bias

The bias of the bagged trees is the same as that of the individual trees as the trees are identically distributed

$$E(\hat{y} - y) = E\left(\frac{1}{B}\sum_{b=1}^{B}(\hat{y}_b - y)\right) = \frac{1}{B}\sum_{b=1}^{B}E(\hat{y}_b - y) = E(\hat{y}_b - y)$$

Bagging - Variance

The variance of the average of the bagged estimates is (ρ is the correlation between pairs of trees)

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

The second term goes to zero as B increases. We see that the size of the correlation between pairs of bagged trees, ρ, is what limits the variance reduction.

Bagging

- Particularly good for high-variance, low-bias methods such as trees
- **Regression**
  - Regression trees are fitted to bootstrap samples of the training data. The result is the average over all of the trees.
- **Classification**
  - A committee of trees each cast a vote for the class - and the majority vote is used as the prediction.

Questions?