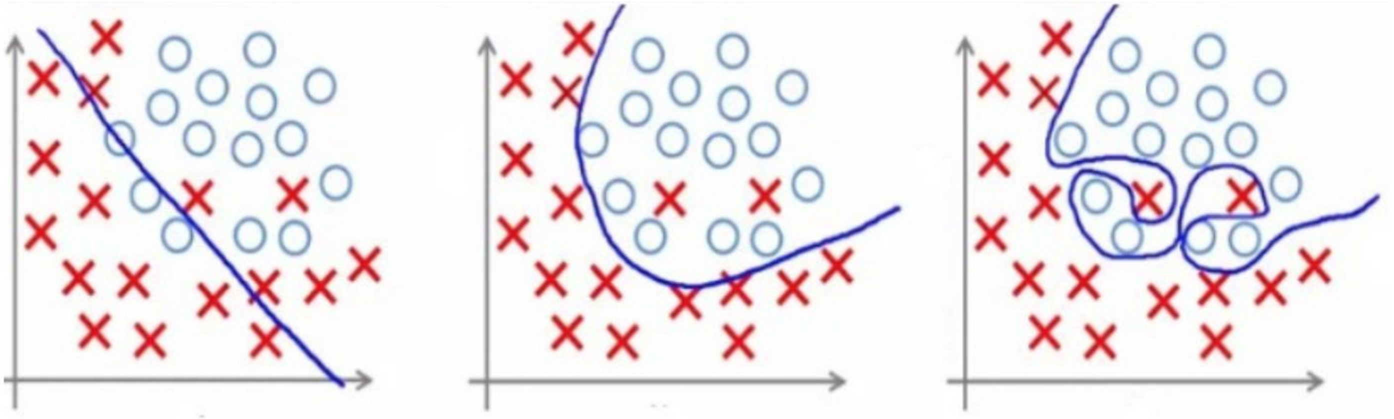
Model Overfitting

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

What overfitting looks like?



Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Source: https://towardsdatascience.com/a-visual-look-at-under-and-overfitting-using-u-s-states-7fd0d8ade053

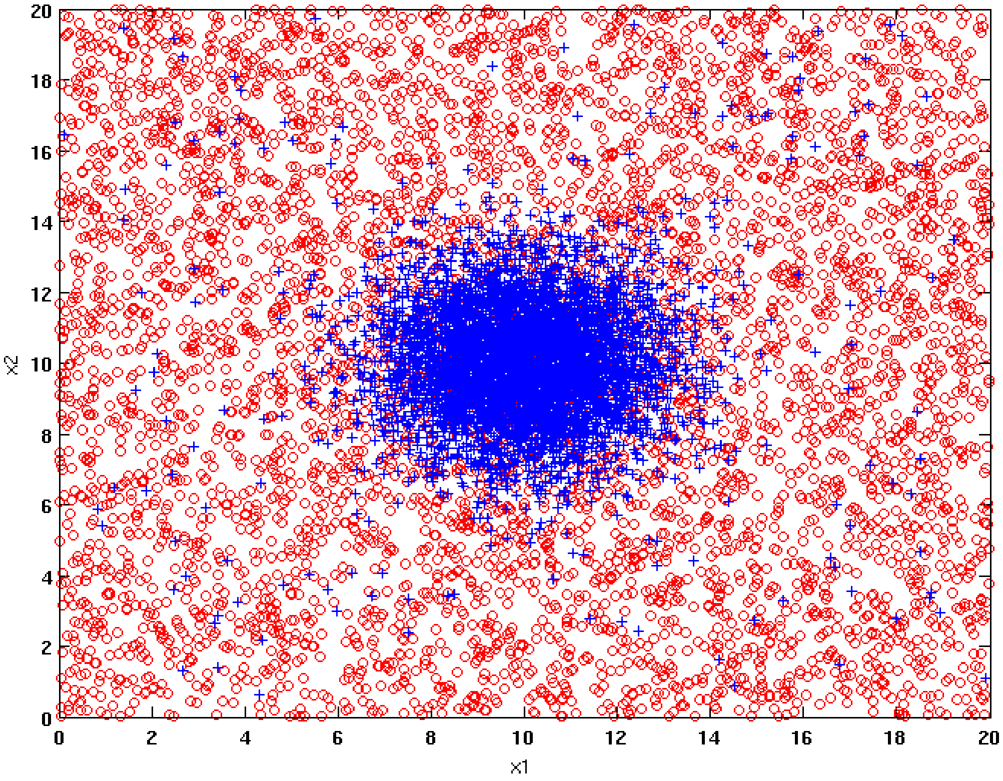
Classification Errors

- Training errors (apparent errors)
  - Errors committed on the training set

- Test errors
  - Errors committed on the test set

- Generalization errors
  - Expected error of a model over random selection of records from same distribution

Example Data Set



Two class problem:

+ : 5400 instances

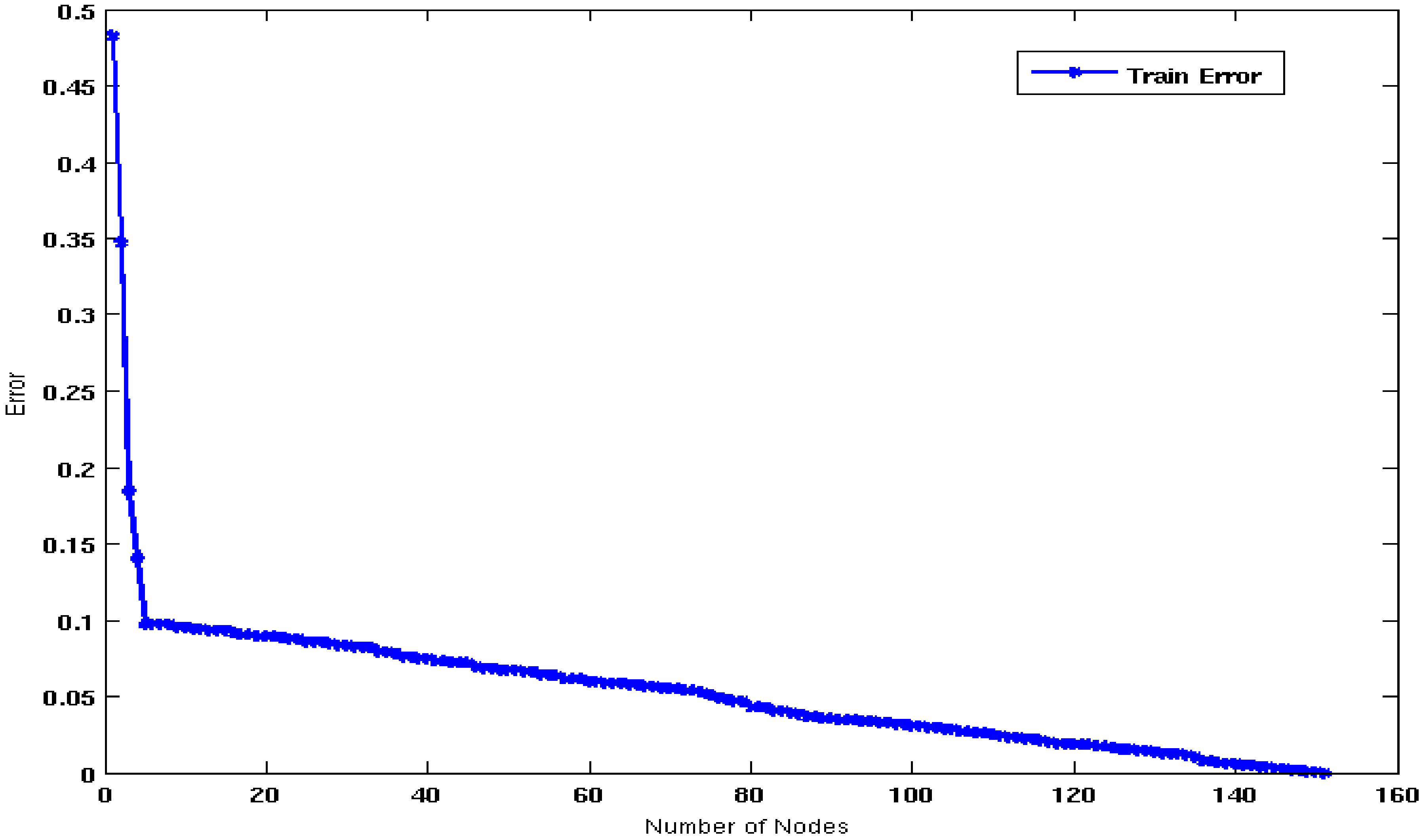
- 5000 instances generated from a Gaussian centered at (10,10)
- 400 noisy instances added

o : 5400 instances

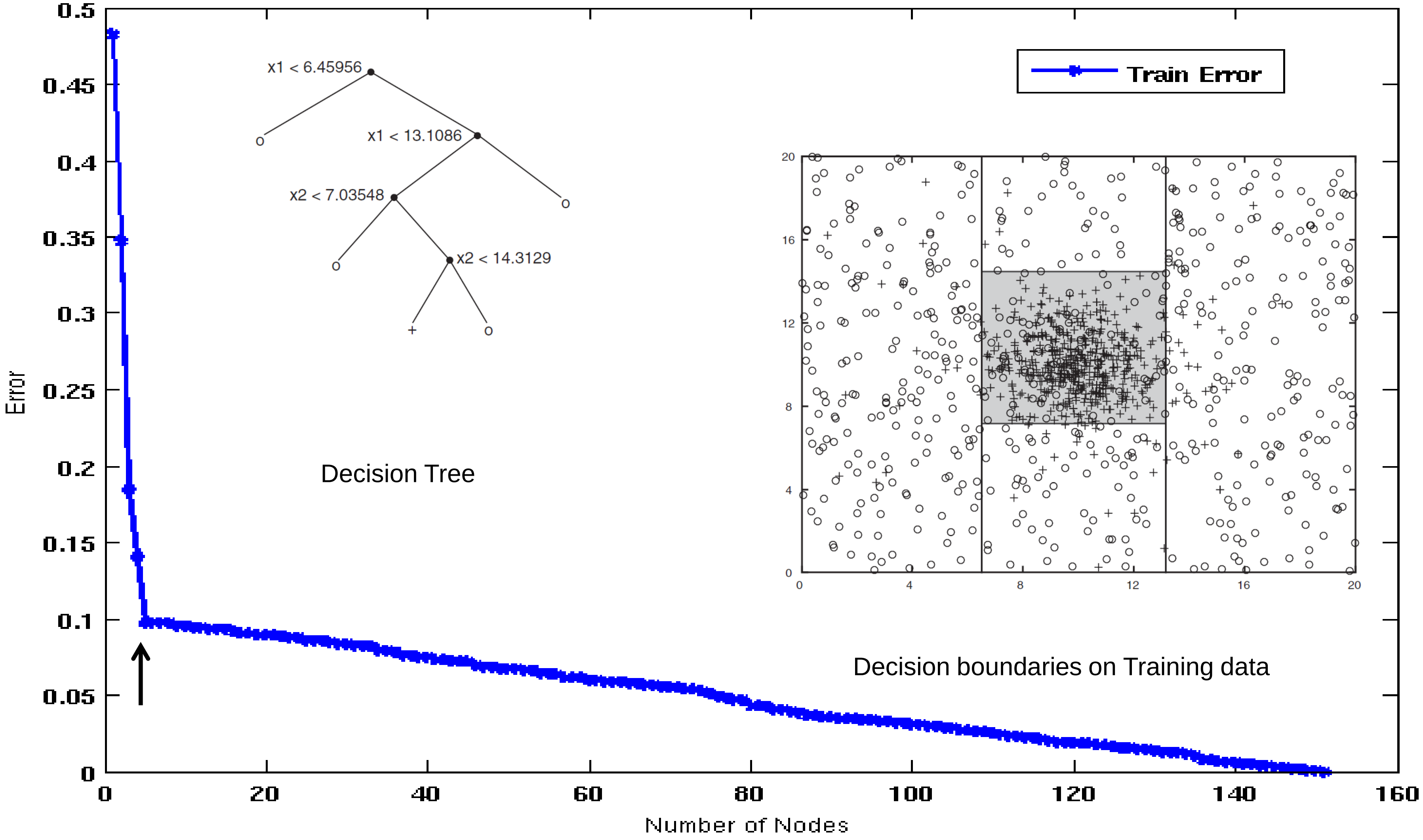
- Generated from a uniform distribution

10 % of the data used for training and 90% of the data used for testing

Increasing number of nodes in Decision Trees



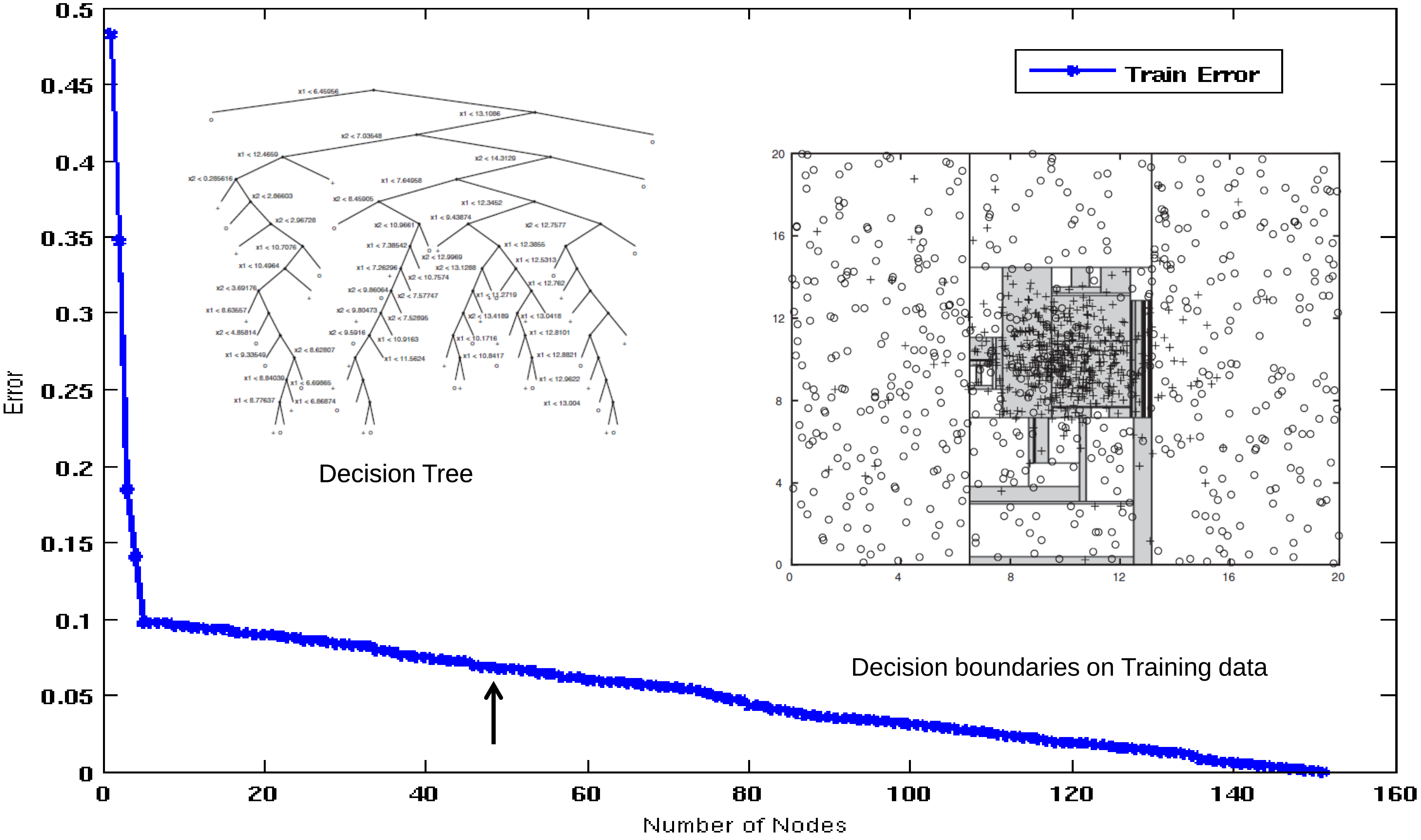
Decision Tree with 4 nodes



Decision Tree

Decision boundaries on Training data

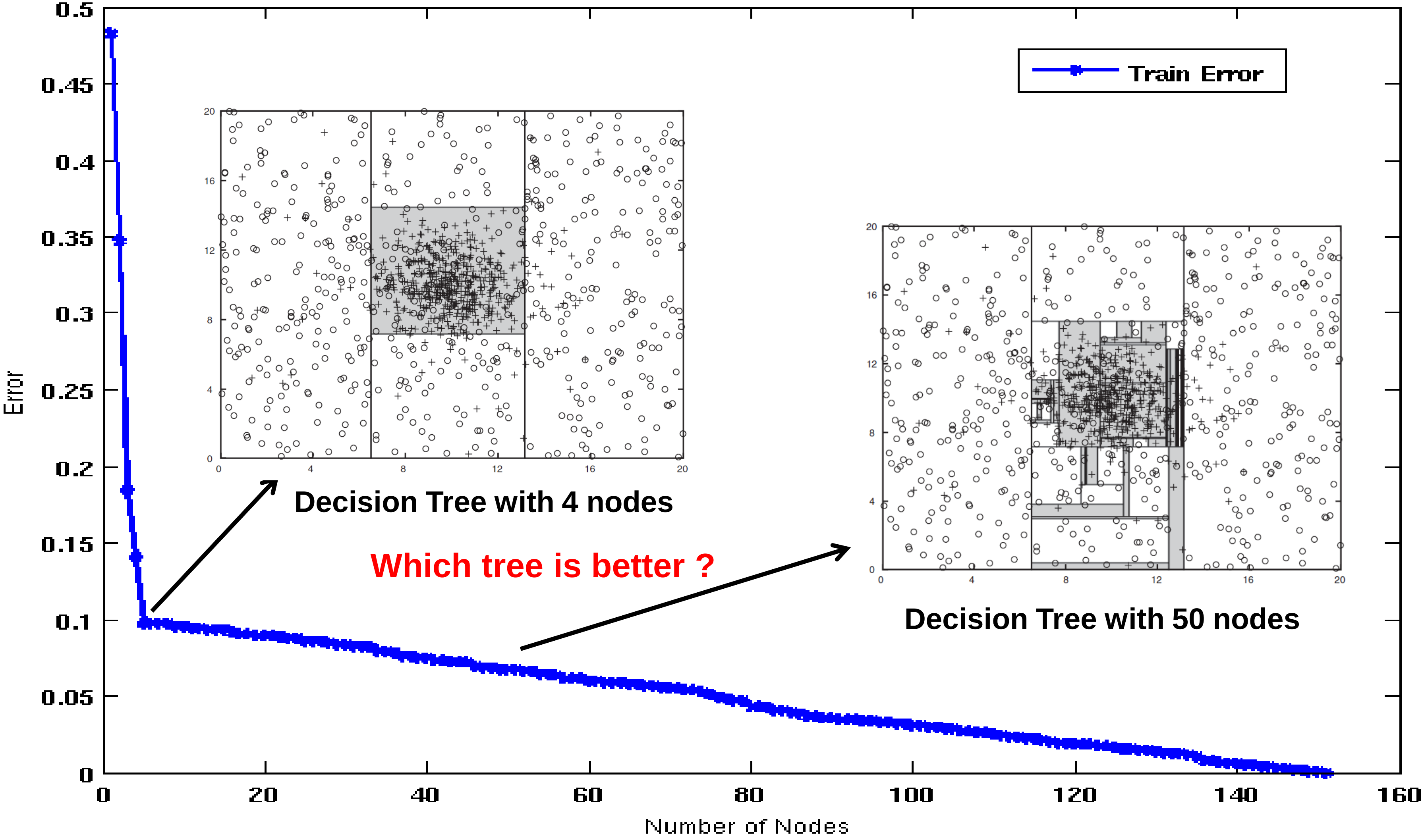
Decision Tree with 50 nodes



Decision Tree

Decision boundaries on Training data

Which tree is better?

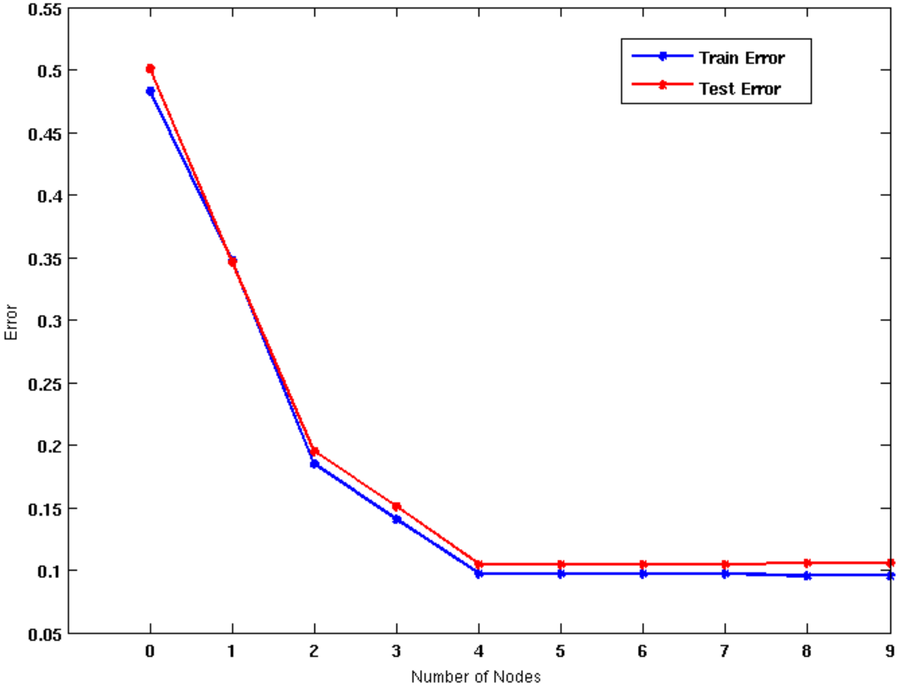


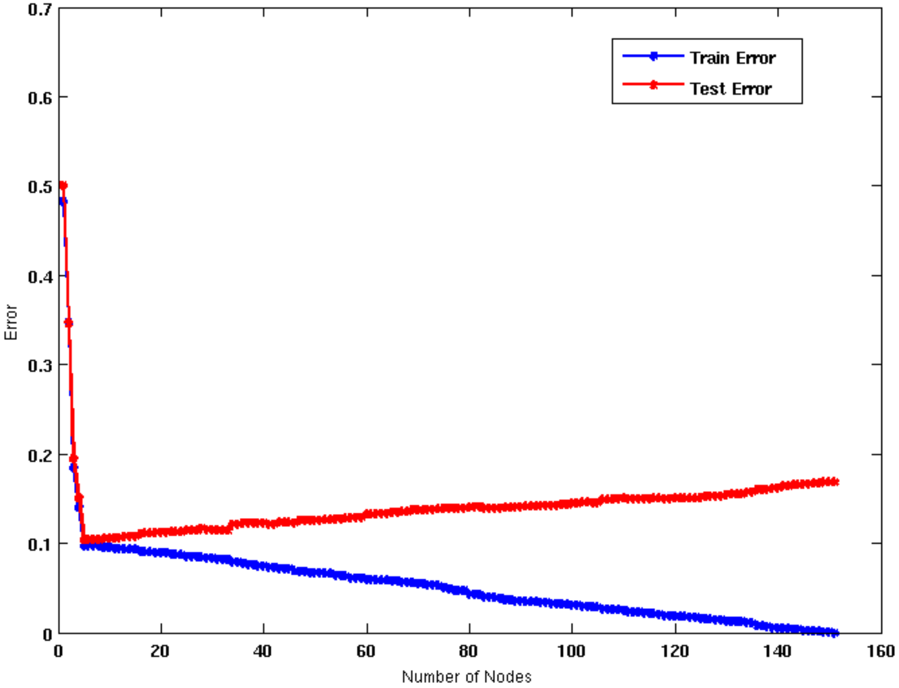
Decision Tree with 4 nodes

Which tree is better ?

Decision Tree with 50 nodes

Model Overfitting



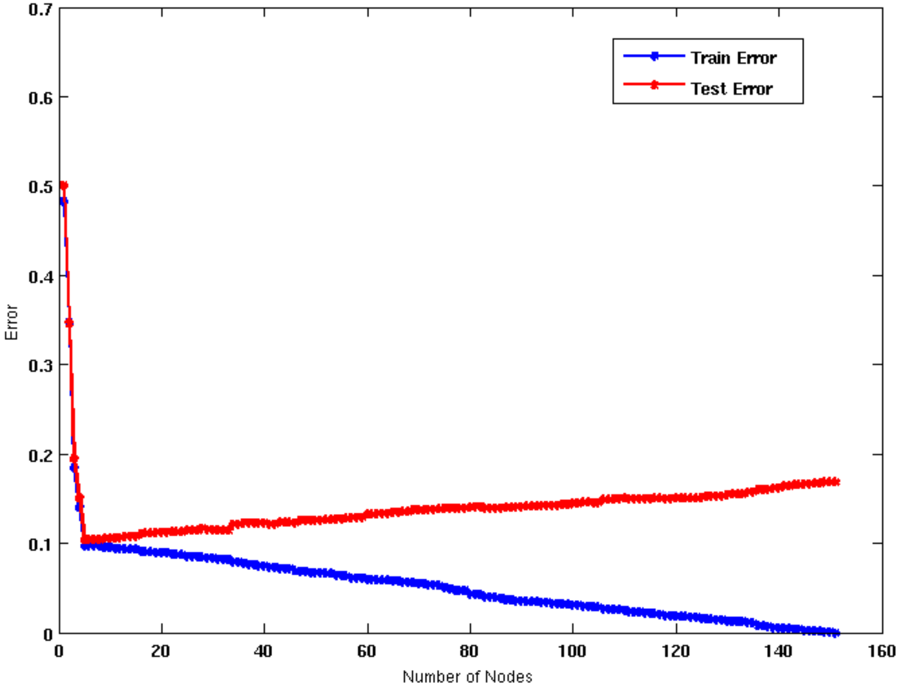


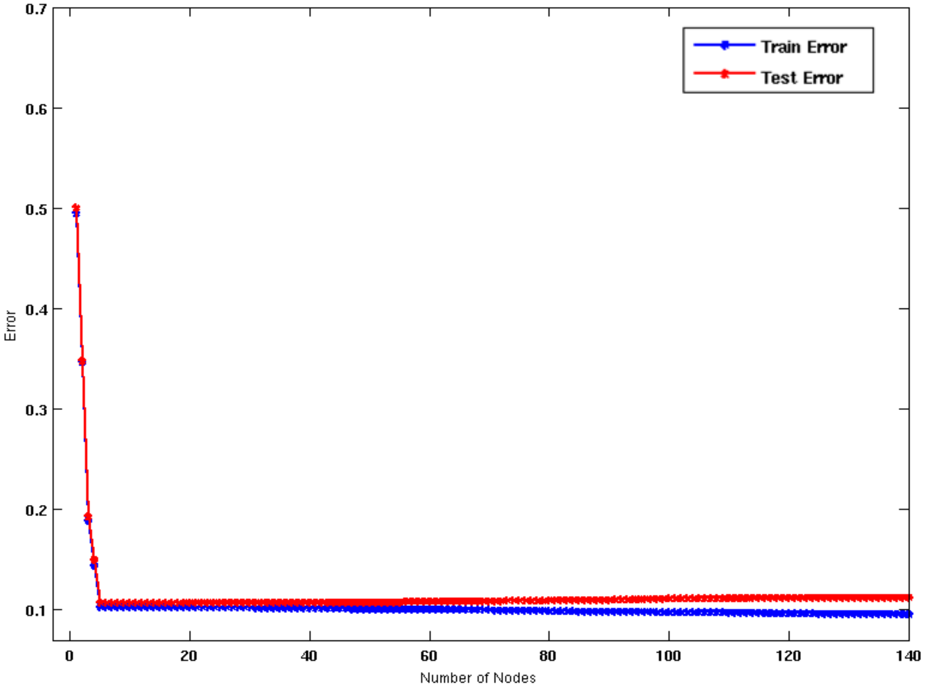
- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

Underfitting: when model is too simple, both training and test errors are large

Overfitting: when model is too complex, training error is small but test error is large

Model Overfitting

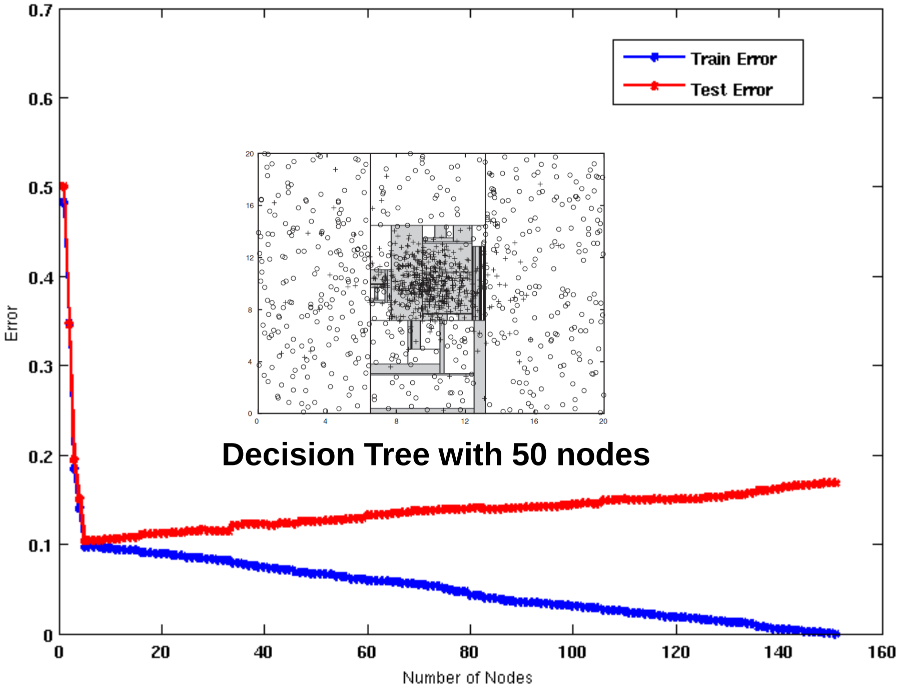




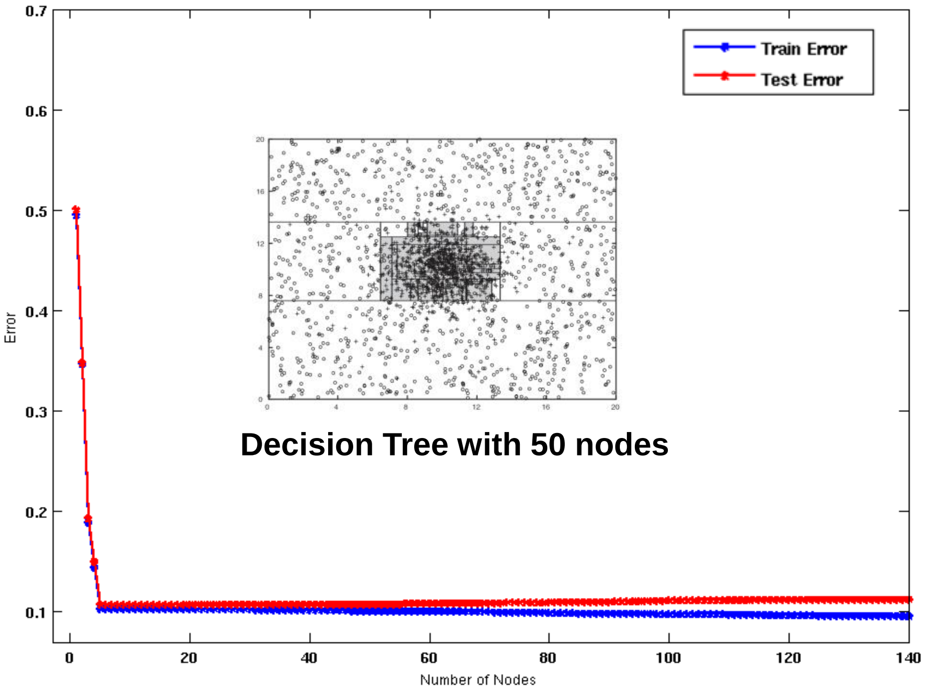
Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Model Overfitting



Decision Tree with 50 nodes

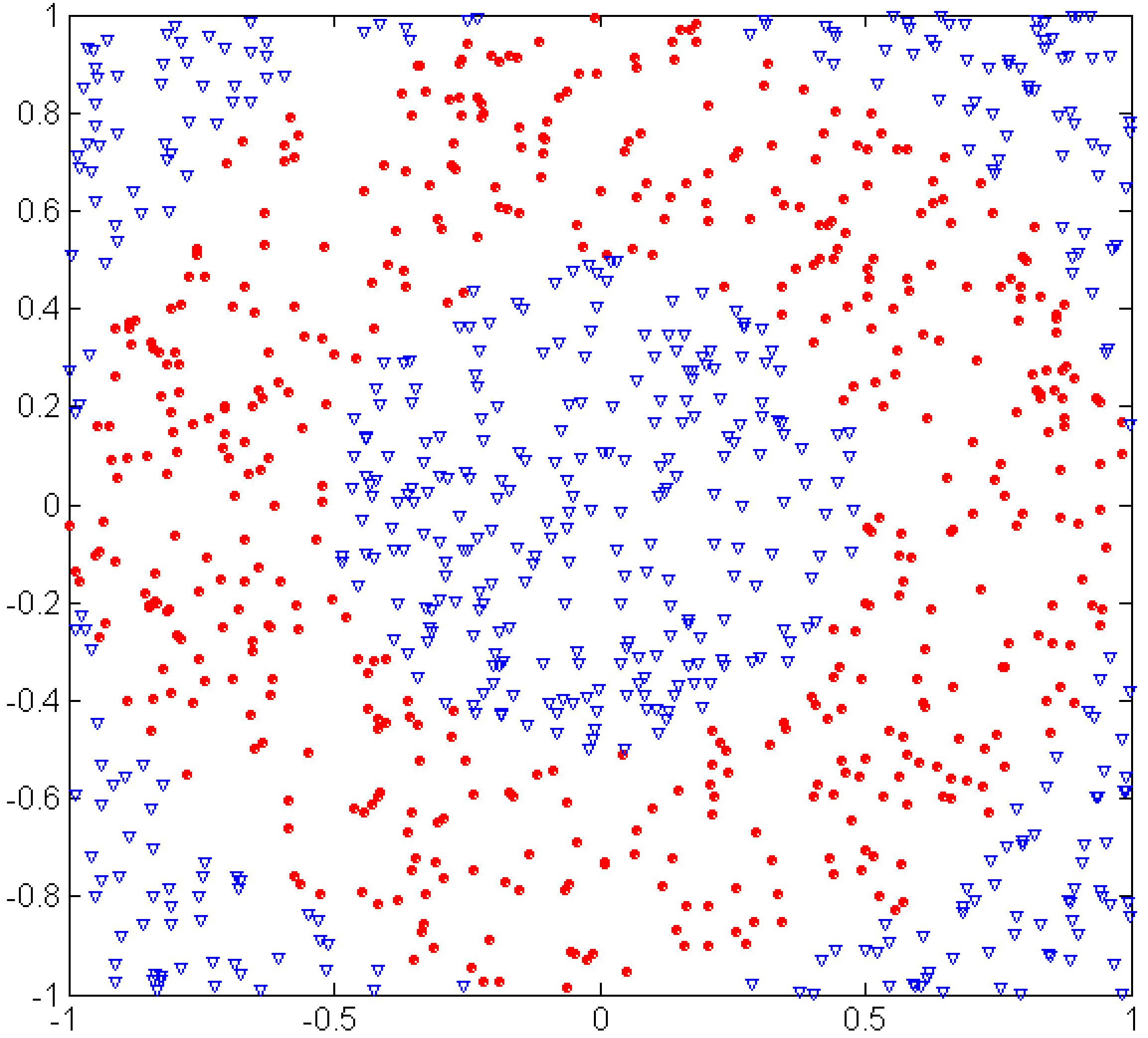


Decision Tree with 50 nodes

Using twice the number of data instances

- Increasing the size of training data reduces the difference between training and testing errors at a given size of model

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

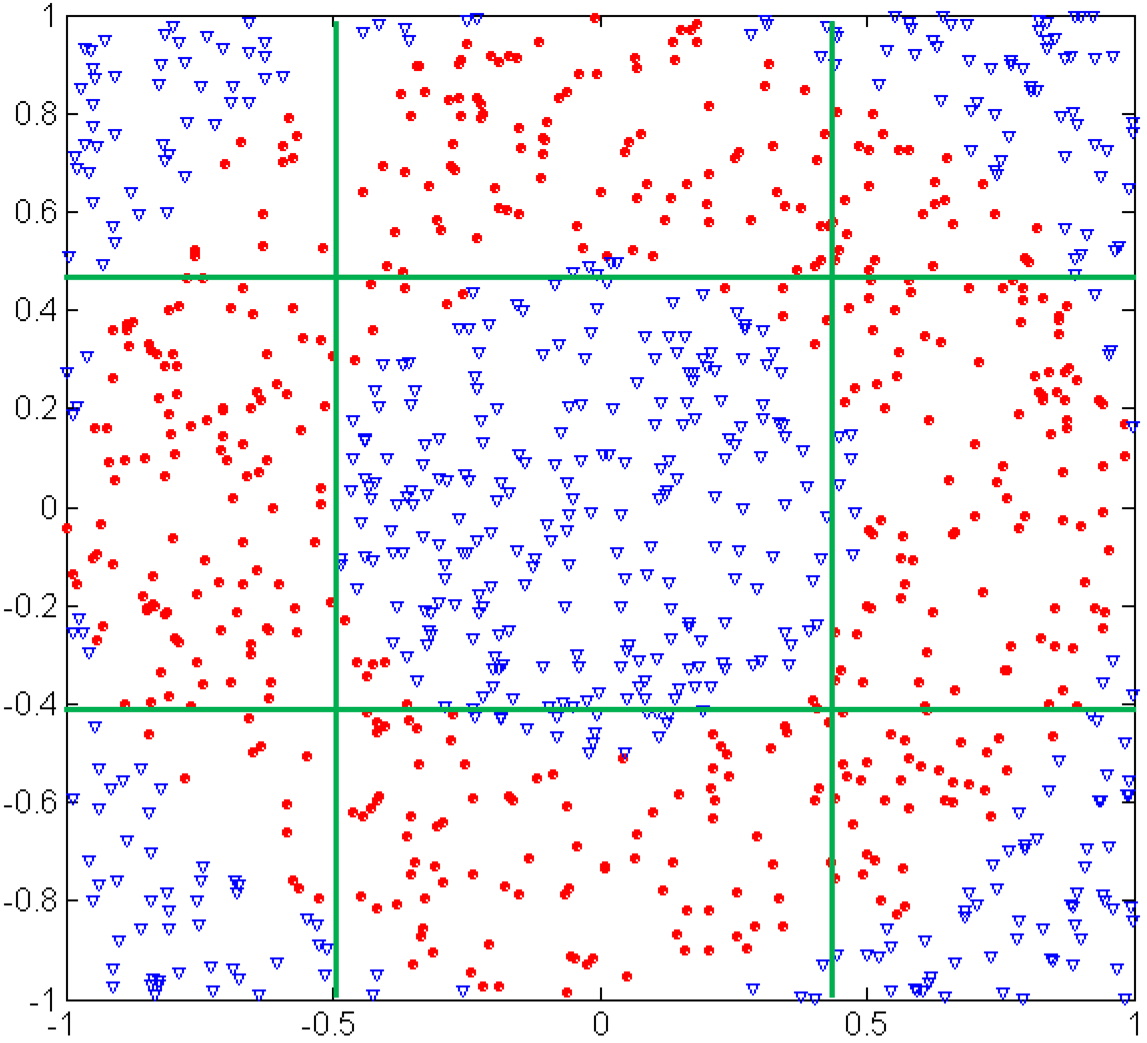
$0.5 \leq \mathrm{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\mathrm{sqrt}(x_1^2+x_2^2) < 0.5$ or

$\mathrm{sqrt}(x_1^2+x_2^2) > 1$

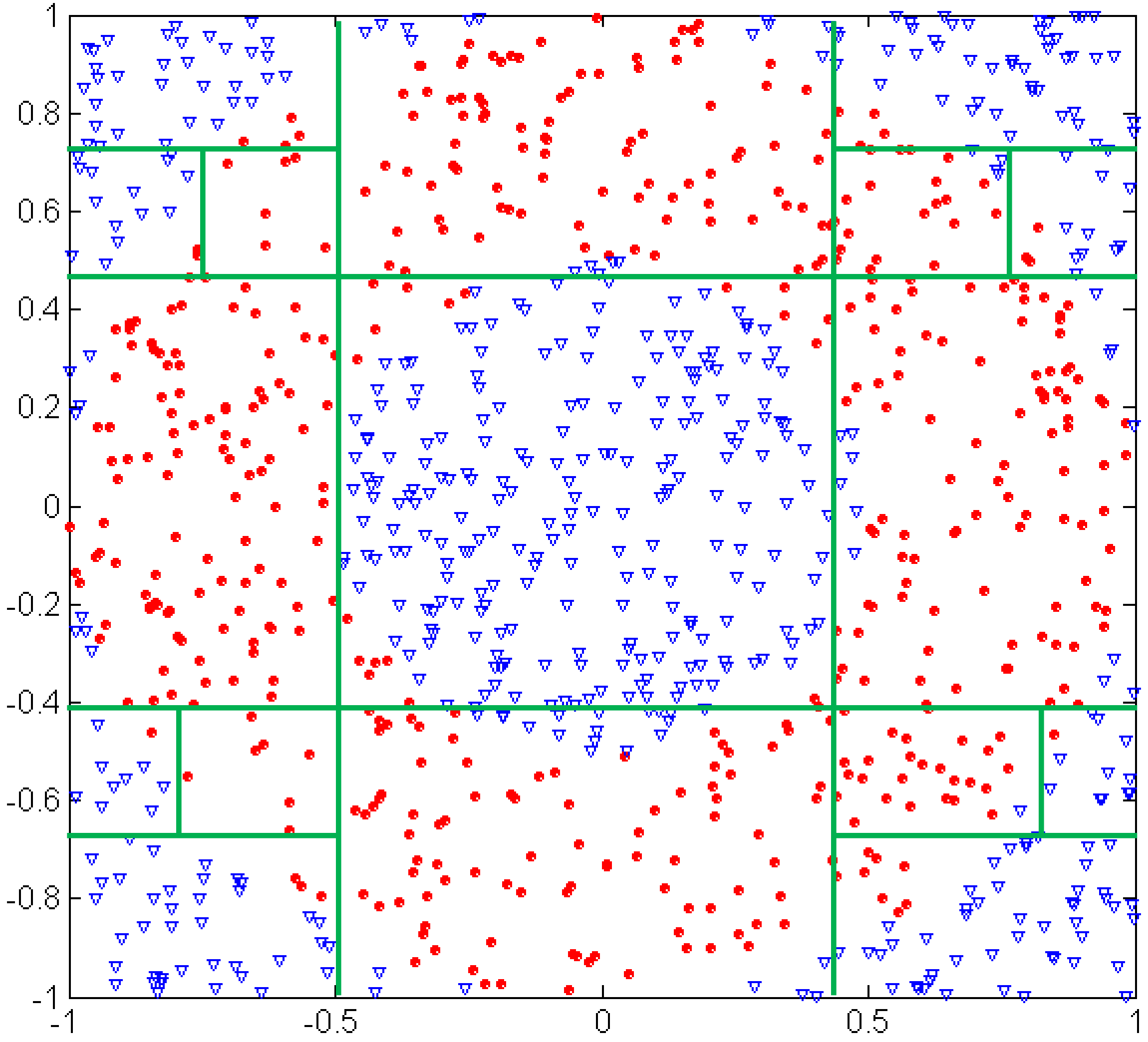
Underfitting and Overfitting (Example)



Decision tree uses lines parallel to the axis to split the space.

Can you draw some such lines to split the space to minimize the error on the training data?

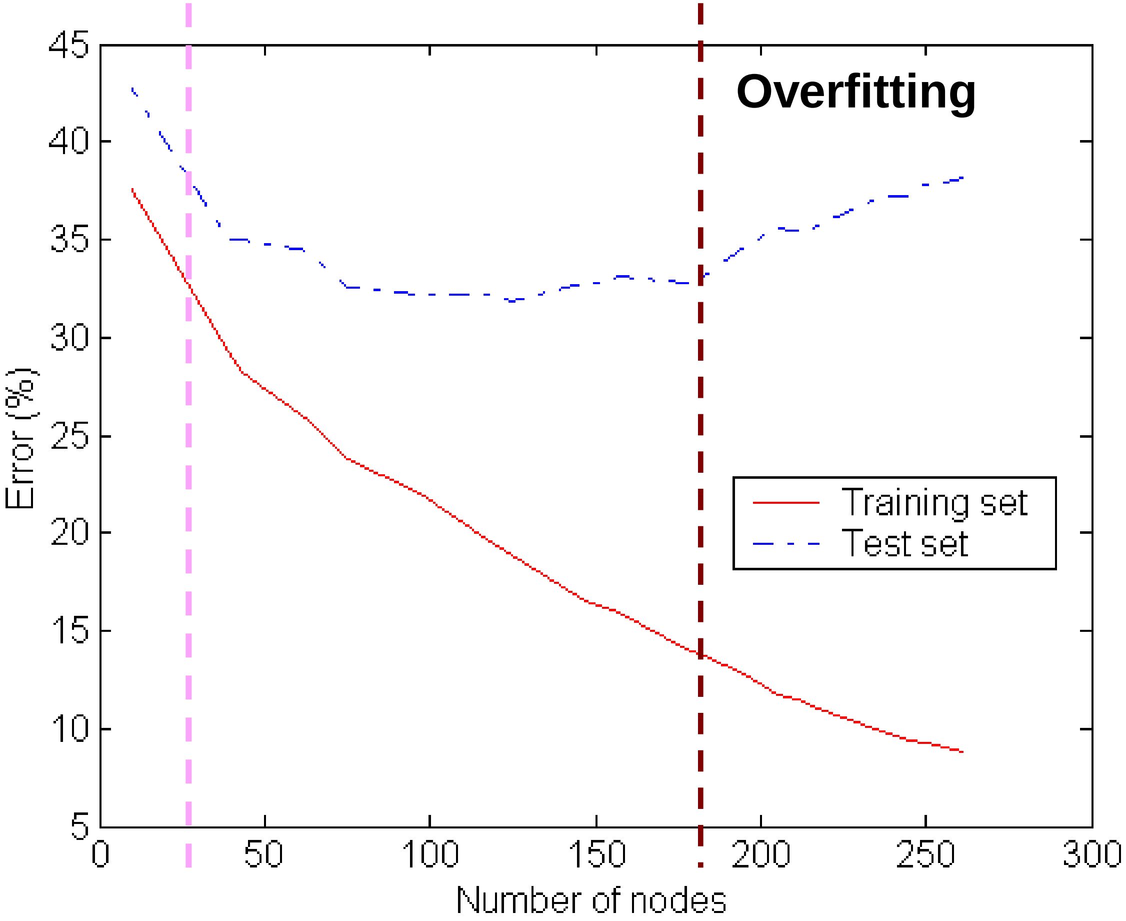
Underfitting and Overfitting (Example)



Does more splitting on training data mean improved prediction rate on test data?

(test=validation here)

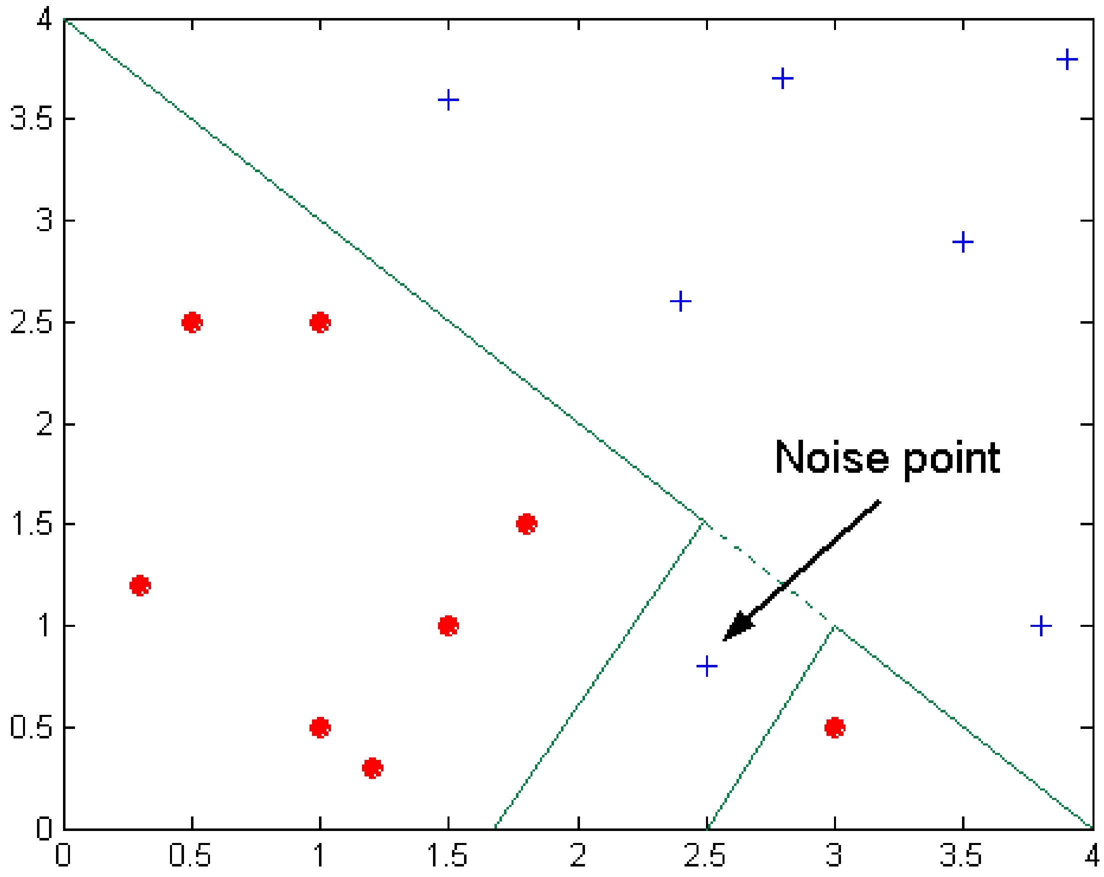
Underfitting and Overfitting



Overfitting due to:
- **Noise**
- **Insufficient examples**

Underfitting: when model is too simple, both training and test errors are large

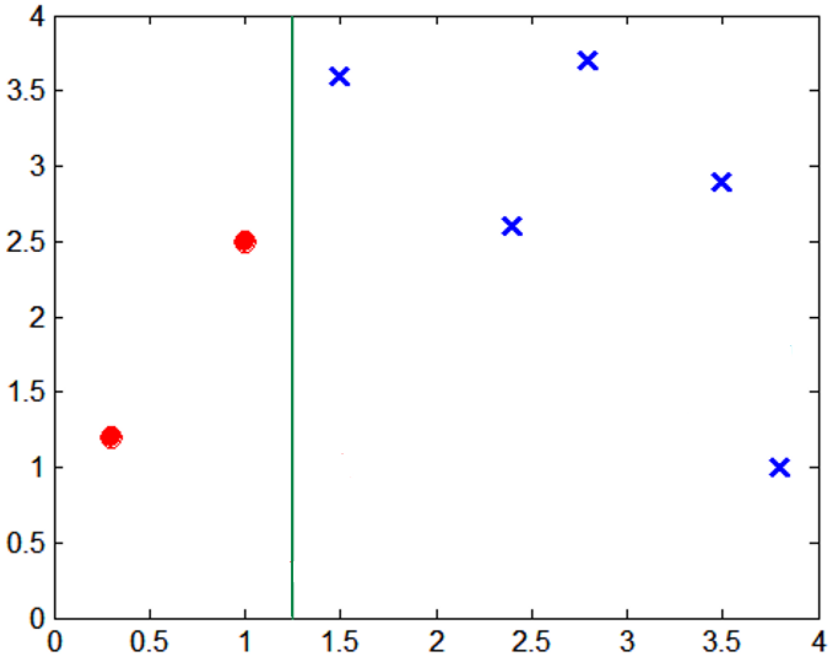
Overfitting due to Noise



The area should be 'o' while overfitting makes it '+'

Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples

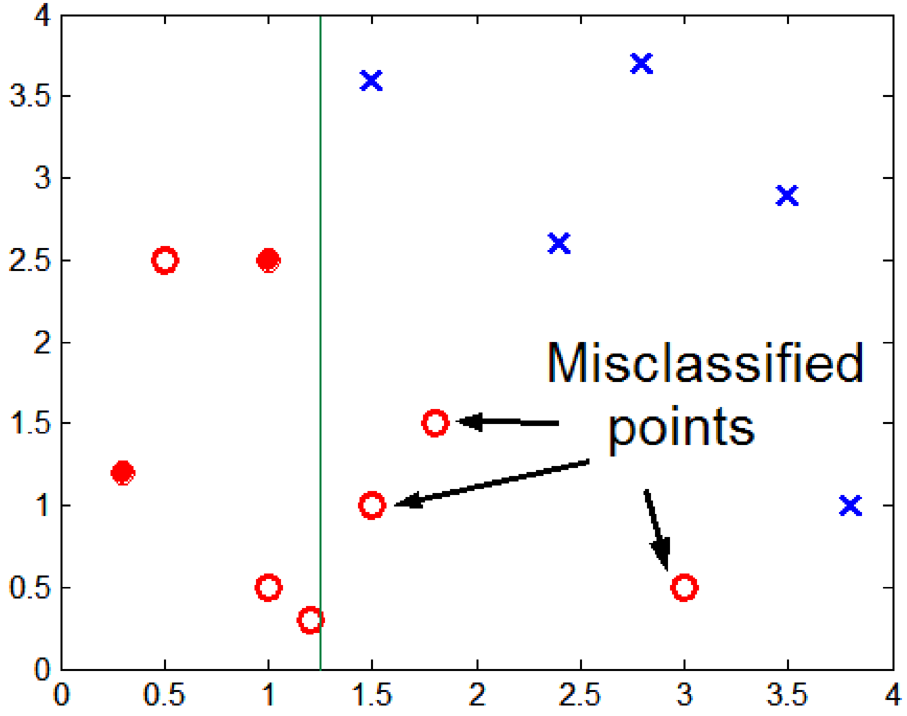


A split based on very few points of two classes red and blue.

Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Overfitting due to Insufficient Examples



Circles are test points all red.

Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Reasons for Model Overfitting

- Limited Training Size

- High Model Complexity
  - Multiple Comparison Procedure

Effect of Multiple Comparison Procedure

- Consider the task of predicting whether stock market will rise/fall in the next 10 trading days

- Random guessing:

  $P(correct) = 0.5$

- Make 10 random guesses in a row:

$$P(\# correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

Effect of Multiple Comparison Procedure

- Approach:
  - Get 50 analysts
  - Each analyst makes 10 random guesses
  - Choose the analyst that makes the most number of correct predictions

- Probability that at least one analyst makes at least 8 correct predictions

$$P(\#\,correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

Effect of Multiple Comparison Procedure

- Many algorithms employ the following greedy strategy:
  - Initial model: M
  - Alternative model: M’ = M $\cup$ γ, where γ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep M’ if improvement, Δ(M,M’) > α

- Often times, γ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

- If many alternatives are available, one may inadvertently add irrelevant components to the model, resulting in model overfitting

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary

- Training error does not provide a good estimate of how well the tree will perform on previously unseen records

- Need ways for estimating generalization errors

Estimating Generalization Errors

- Use Resubstitution Estimate
  - Optimistic approach: e’(t) = e(t)

- Why this might not be the best approach?

Estimating Generalization Errors

- Use Resubstitution Estimate
  - Optimistic approach: $e'(t) = e(t)$

- Incorporating Model Complexity
  - Occam’s Razor – we should prefer simpler models

Occam’s Razor – a principle

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model

- For complex models, there is a greater chance that it was fitted accidentally by errors in data

- Therefore, one should include model complexity when evaluating a model

Estimating Generalization Errors

- Use Resubstitution Estimate
  - Optimistic approach: e'(t) = e(t)

- Incorporating Model Complexity
  - Occam's Razor – we should prefer simpler models
  - "Everything should be made as simple as possible, but not simpler"
  - Two models:
    - Pessimistic error estimate
    - Maximum description length principle

Pessimistic Error Estimate

- The sum of training error and a penalty term for model complexity

$$e_g(T) = \frac{e(T) + \Omega(T)}{N_t}$$

$e(T)$	the overall training error
$\Omega(t_i)$	the penalty associated with each node t_i
N_t	The number of training records

Pessimistic Error Estimate

- The sum of training error and a penalty term for model complexity

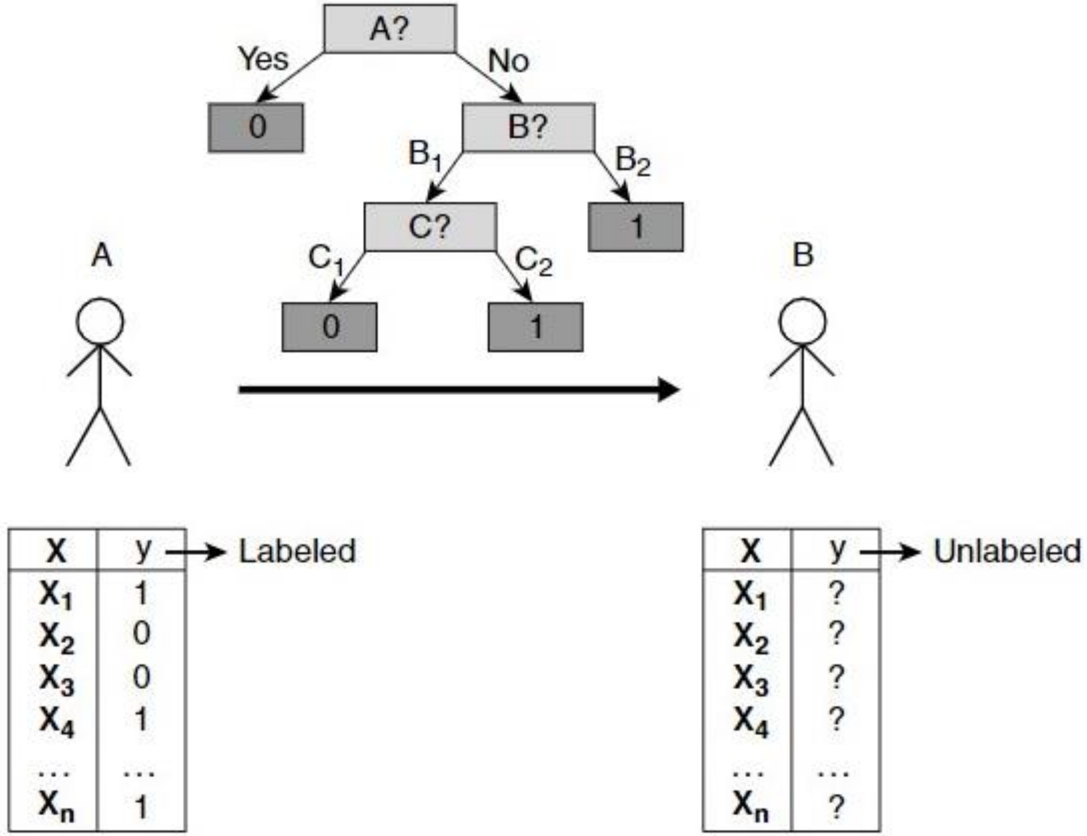
$$e_g(T) = \frac{e(T) + \Omega(T)}{N_t}$$

$e(T)$	the overall training error
$\Omega(t_i)$	the penalty associated with each node t_i
N_t	The number of training records

$\Omega(t)$ = 0.5 – a node should be expanded into its two child nodes as long as it improves the classification of **at least one training record.**

$\Omega(t)$ = 1 – a node should **NOT** be expanded into its child nodes unless it reduces the misclassification error for **more than one training record.**

Minimum Description Length (MDL)

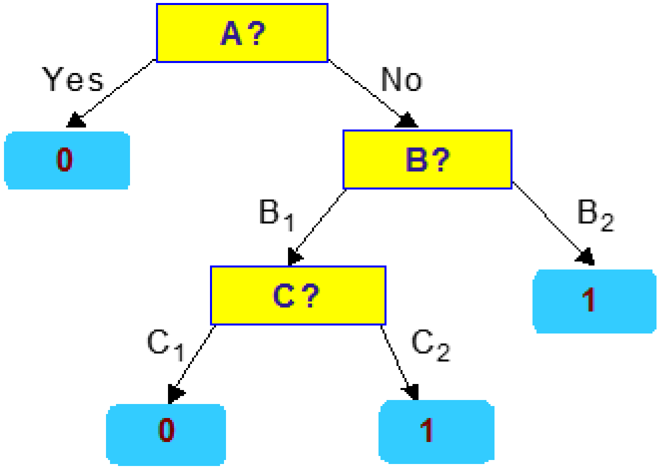


X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1

X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

Minimum Description Length (MDL)

Dilemma:
less errors + complex tree
vs
more errors + simple tree



larger when tree has more errors

larger when tree has more nodes

- Cost(Model,Data) = Cost(Data|Model) + Cost(Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

Estimating Generalization Errors

- Use Resubstitution Estimate
  - Optimistic approach: e’(t) = e(t)

- Incorporating Model Complexity
  - Occam’s Razor – we should prefer simpler models
  - “Everything should be made as simple as possible, but not simpler”
  - Two models:
    - Pessimistic error estimate
    - Maximum description length principle

- Using a Validation Set

How to Address Overfitting

- Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node:
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or information gain).

How to Address Overfitting…

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - Trimming can be done by replacing a subtree with:
    - a new leaf node whose class label is determined from the majority class of records affiliated with the subtree
    - the most frequently used branch of the subtree

Example of Post-Pruning

Training Error = 10/30

Pessimistic generalization error

= (10 + 0.5)/30 = 10.5/30

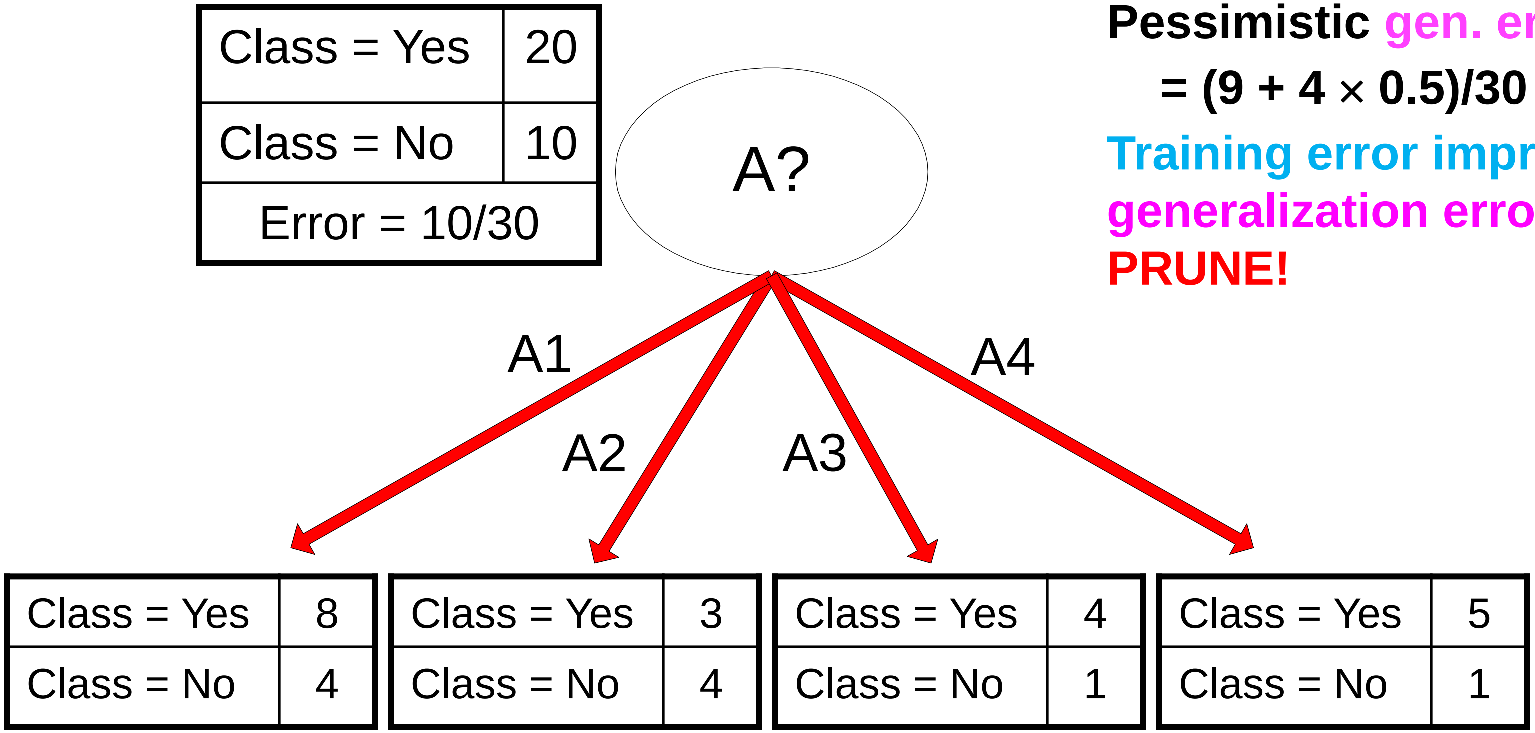
After splitting:

Training Error = 9/30

Pessimistic gen. error

= (9 + 4 × 0.5)/30 = 11/30

Training error improved, but generalization error worsened. PRUNE!



Class = Yes	20
Class = No	10
Error = 10/30	

A1	
Class = Yes	8
Class = No	4

A2	
Class = Yes	3
Class = No	4

A3	
Class = Yes	4
Class = No	1

A4	
Class = Yes	5
Class = No	1

Examples of Post-pruning – case 1

- Follow optimistic error:
  - genErr = train error (trainErr)

e’(parent) = 7/20

e’(children) = 5/20;

7/20 > 5/20: keep

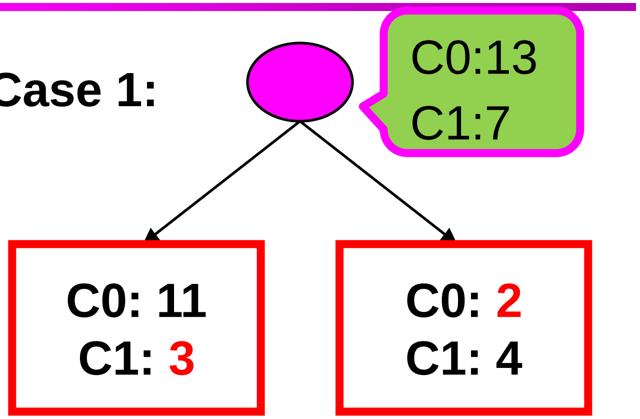
- Follow pessimistic error:
  - genErr = trainErr + nNodes*0.5

e’(parent) = 7.5/20

e’(children) = 6/20

7.5/20 > 6/20: keep.

Case 1:



Examples of Post-pruning – case 2

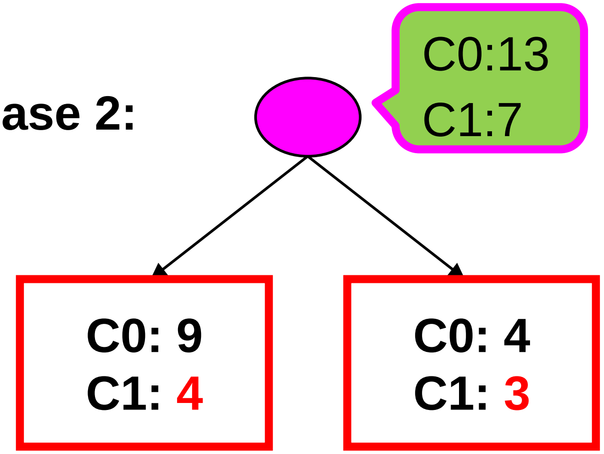
- Follow optimistic error:
  - genErr = train error (trainErr)

e’(parent) = 7/20

e’(children) = 7/20

7/20 > 7/20 is false: prune children

Case 2:



- Follow pessimistic error:
  - genErr = trainErr + nNodes*0.5

e’(parent) = 7.5/20

e’(children) = 8/20

7.5/20 > 8/20 is false: prune children