Data Mining

Model Overfitting

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

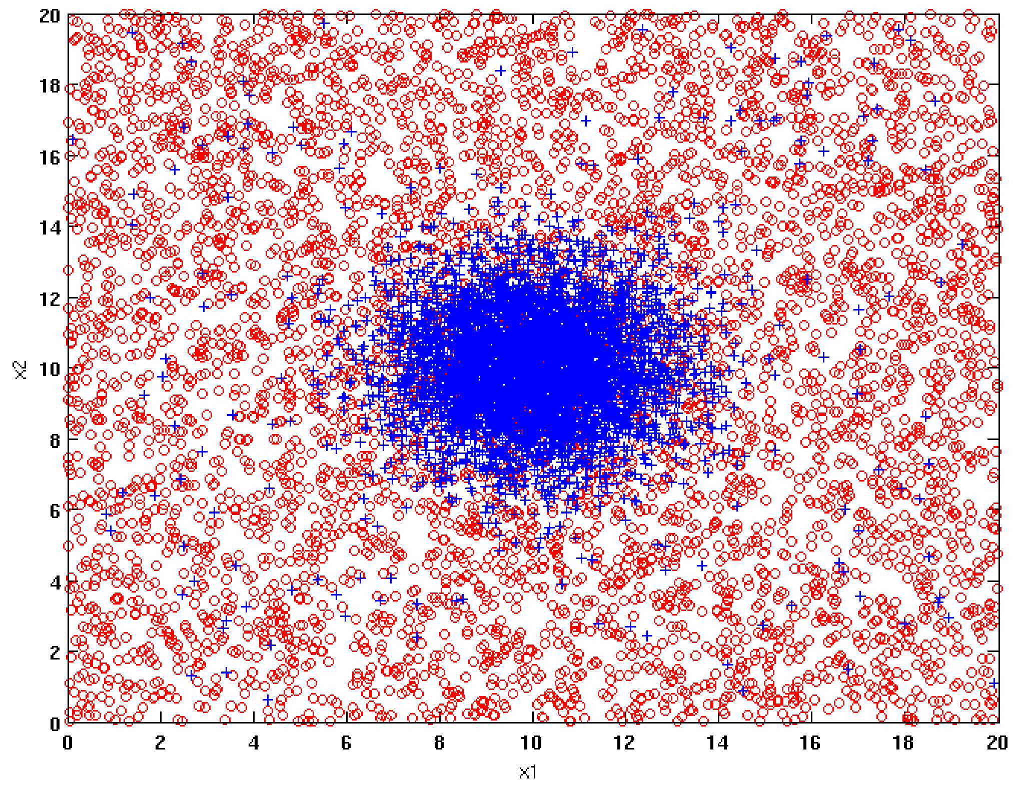
Classification Errors

- 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:

+ : 5200 instances

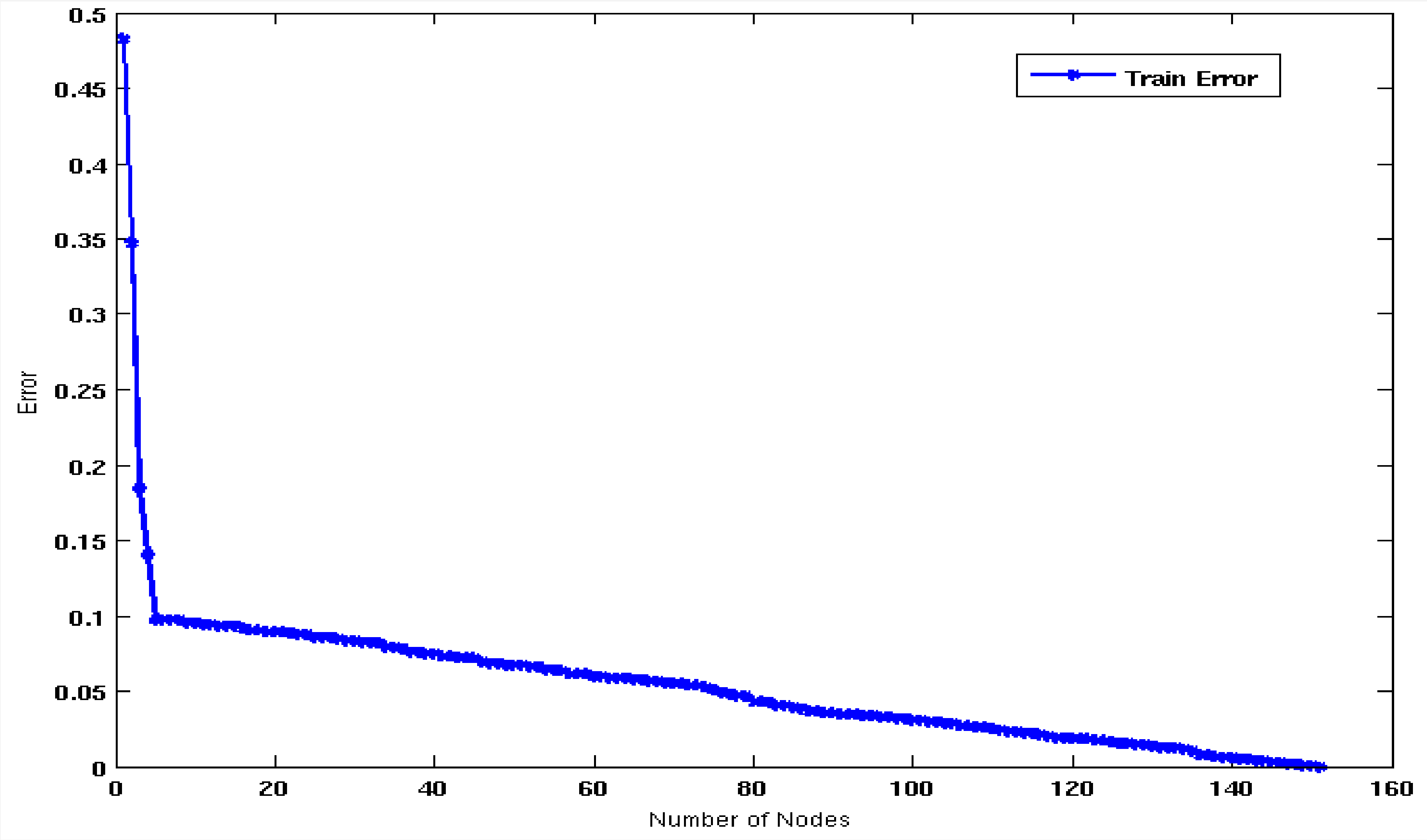
- **5000 instances generated from a Gaussian centered at (10,10)**
- **200 noisy instances added**

o : 5200 instances

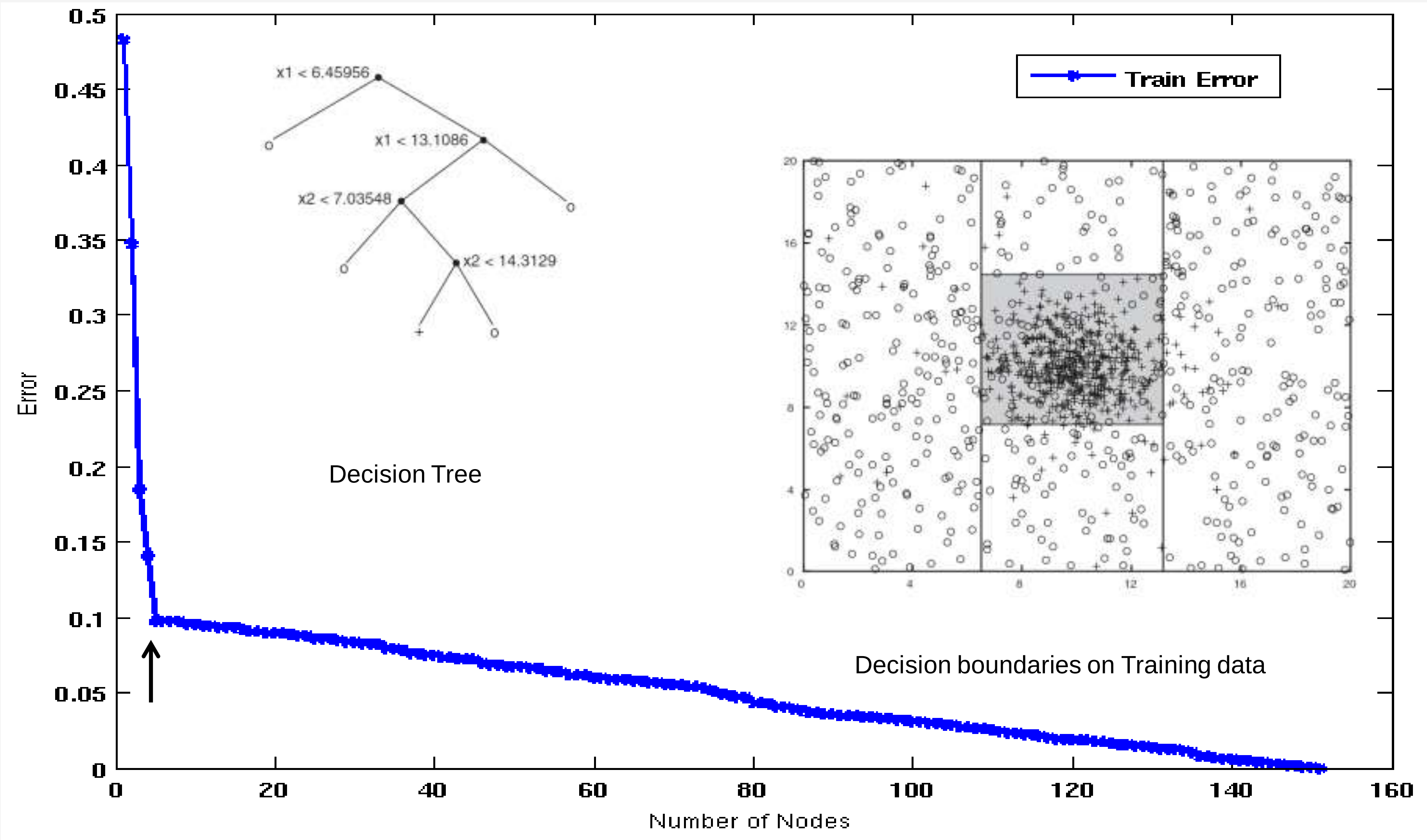
- **Generated from a uniform distribution**

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

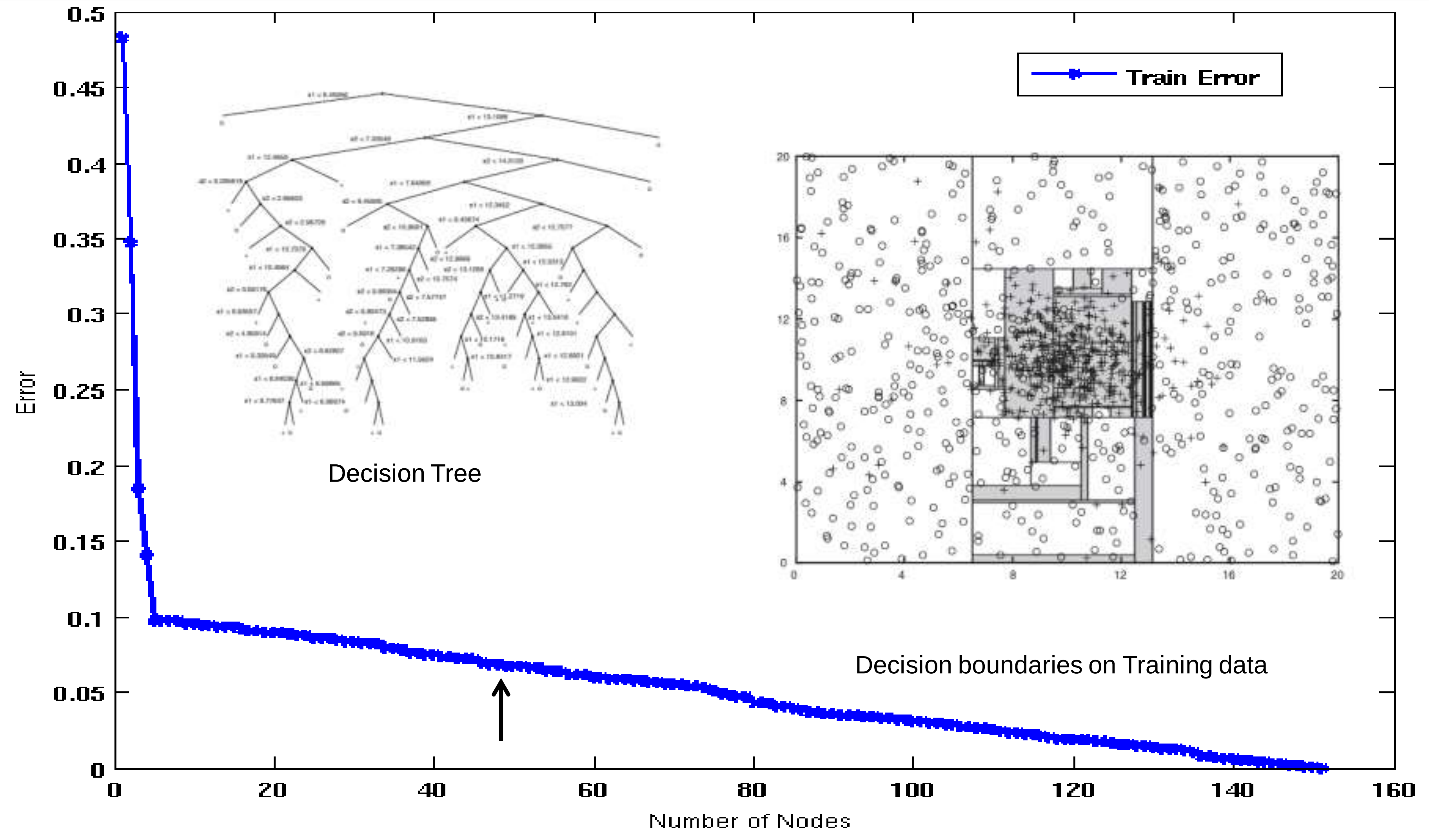
Increasing number of nodes in Decision Trees



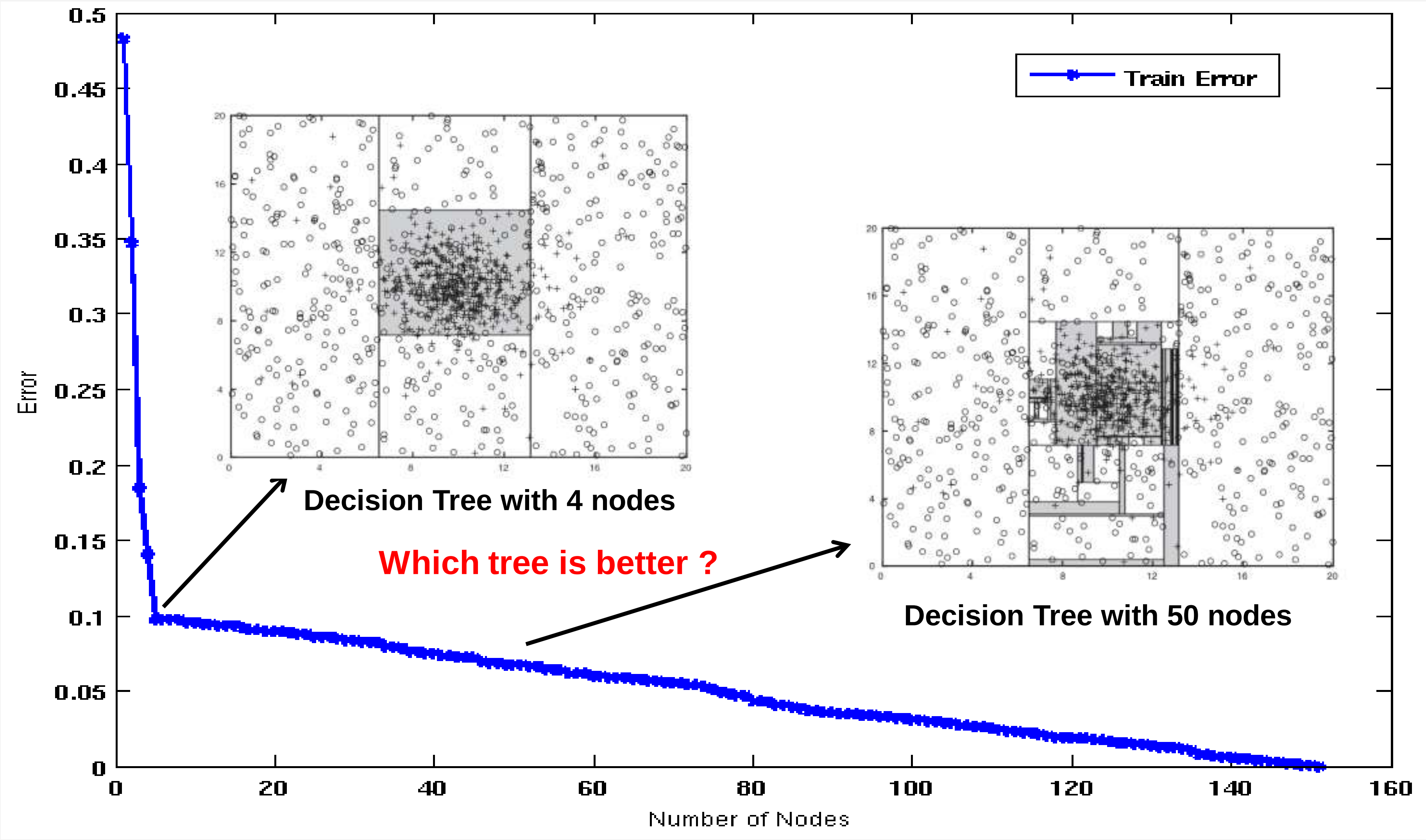
Decision Tree with 4 nodes



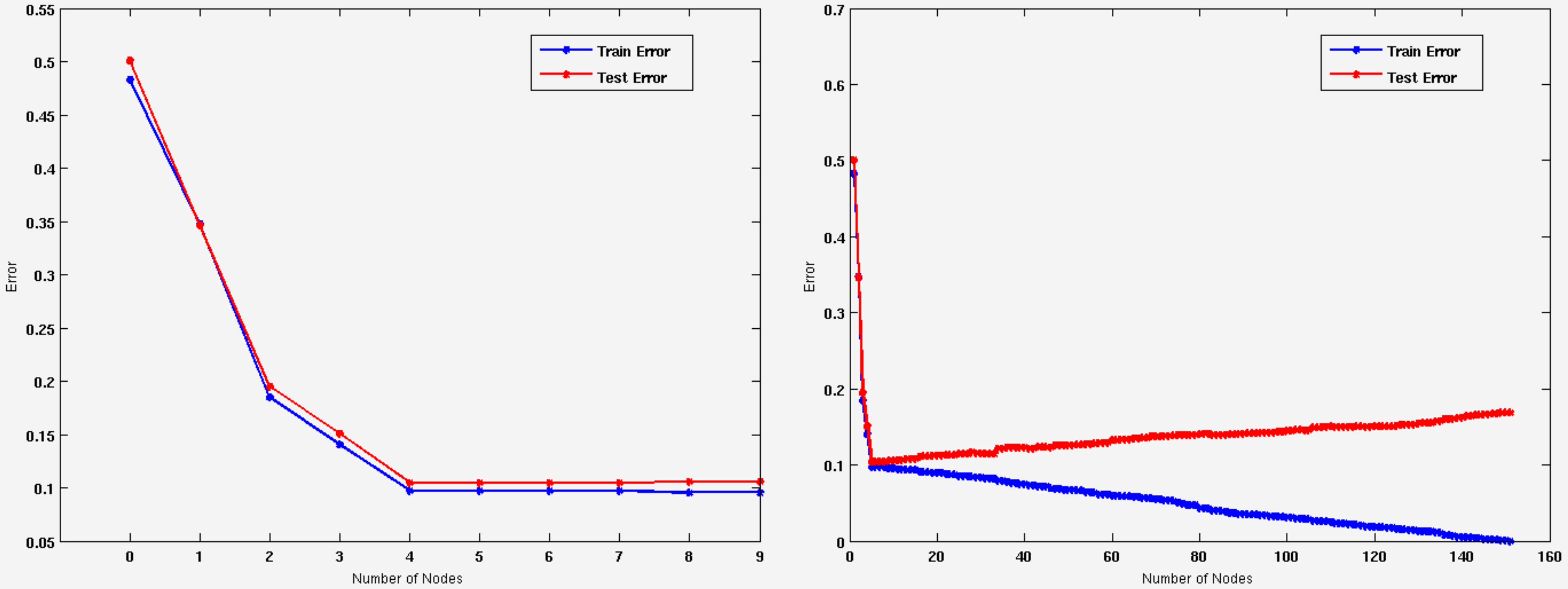
Decision Tree with 50 nodes



Which tree is better?



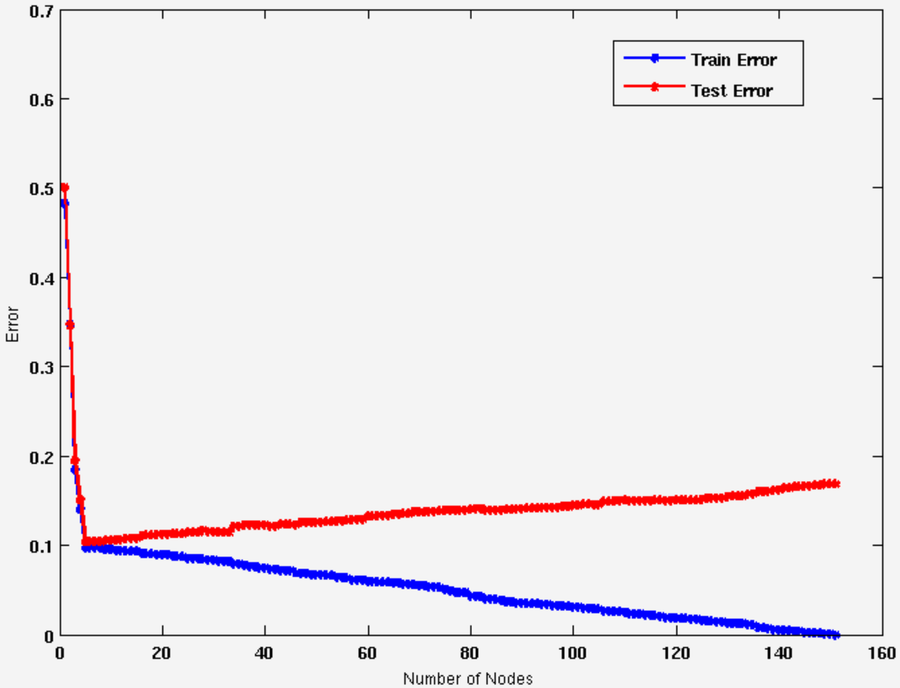
Model Overfitting

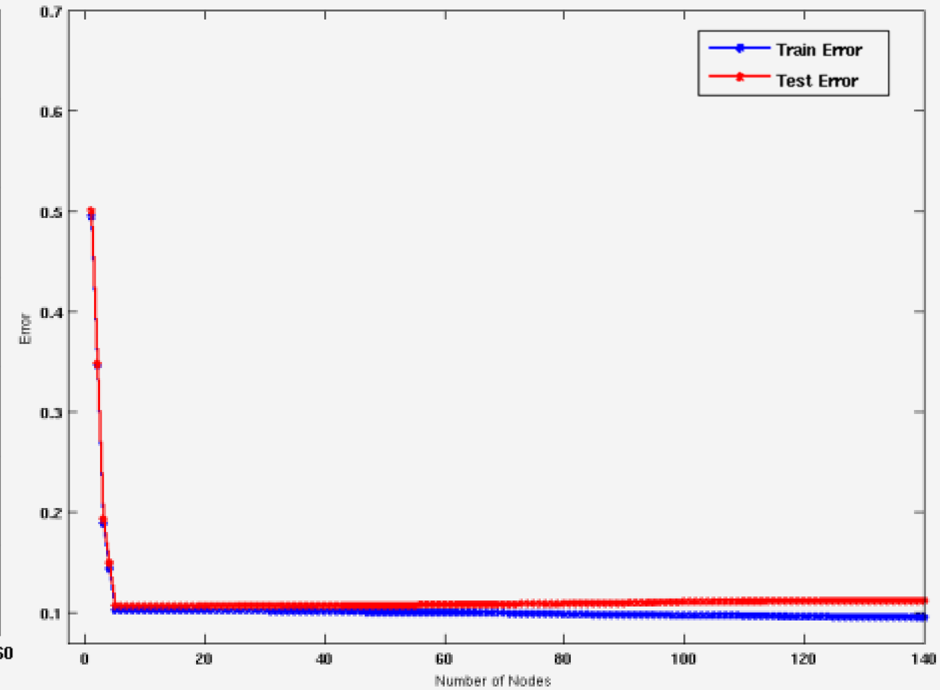


Underfitting: when model is too simple, both training and test errors are large

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

Model Overfitting

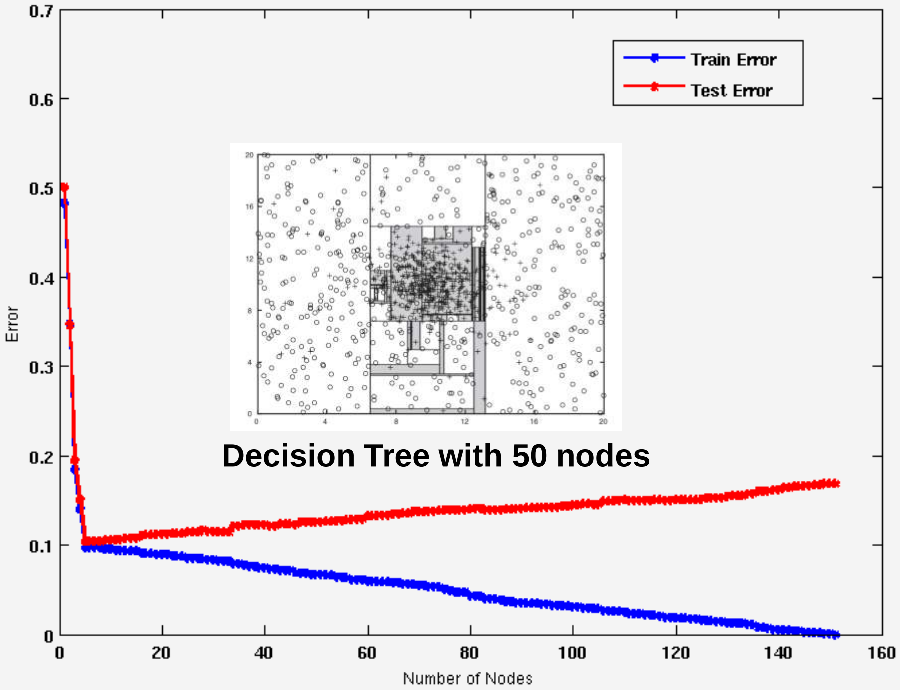


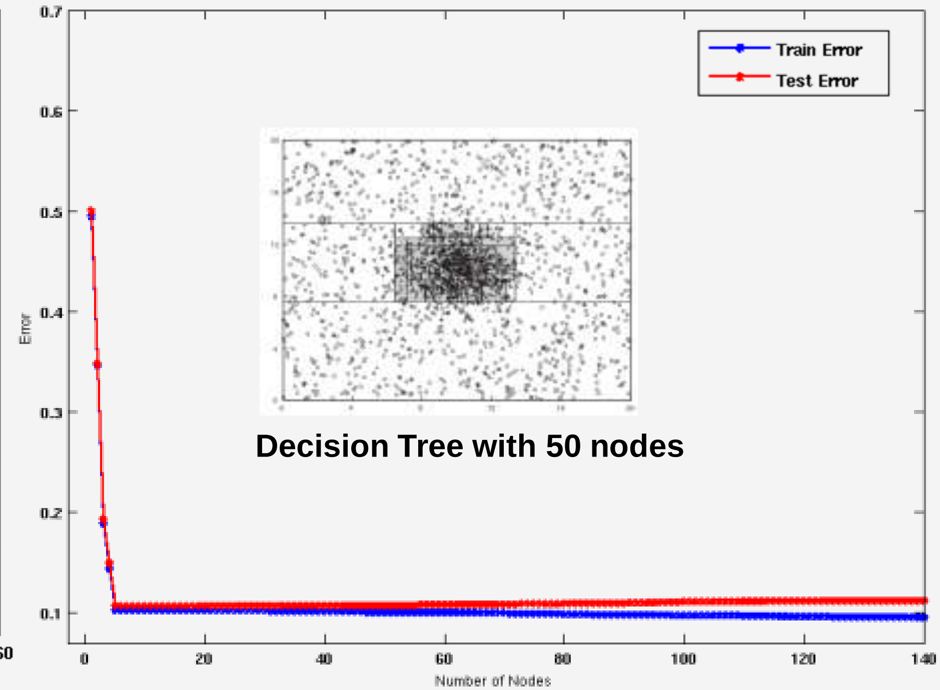


Using twice the number of data instances

- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes

Model Overfitting





Using twice the number of data instances

- If training data is under-representative, testing errors increase and training errors decrease on increasing number of nodes
- Increasing the size of training data reduces the difference between training and testing errors at a given number of nodes

Notes on Overfitting

- Overfitting results in decision trees that are <u>more complex</u> 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

Model Selection

- Performed during model building
- Purpose is to ensure that model is not overly complex (to avoid overfitting)
- Need to estimate generalization error
  - Using Validation Set
  - Incorporating Model Complexity
  - Estimating Statistical Bounds

Model Selection:

Using Validation Set

- Divide <u>training</u> data into two parts:
  - Training set:
    - use for model building
  - Validation set:
    - use for estimating generalization error
    - Note: validation set is not the same as test set

- Drawback:
  - Less data available for training

Model Selection:
Incorporating Model Complexity

- Rationale: Occam's Razor
  - Given two models of similar generalization errors, one should prefer the simpler model over the more complex model
  - A complex model has a greater chance of being fitted accidentally by errors in data
  - Therefore, one should include model complexity when evaluating a model

Gen. Error(Model) = Train. Error(Model, Train. Data) + α x Complexity(Model)

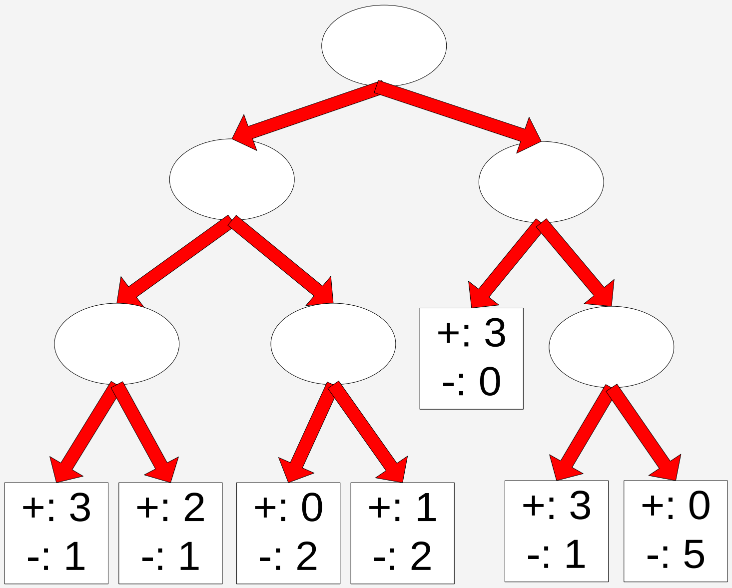
Estimating the Complexity of Decision Trees

- **Pessimistic Error Estimate** of decision tree T with k leaf nodes:

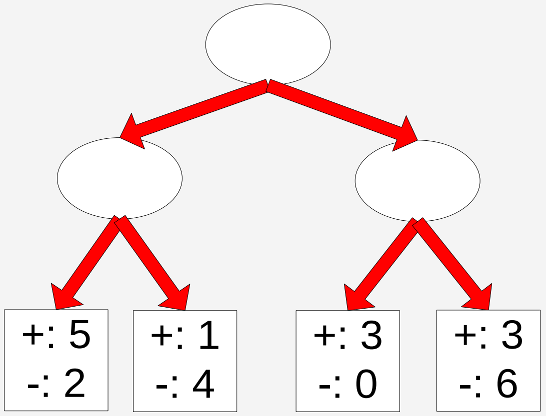
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

  - err(T): error rate on all training records
  - Ω: trade-off hyper-parameter (similar to α)
    - Relative cost of adding a leaf node
  - k: number of leaf nodes
  - N_{train}: total number of training records

Estimating the Complexity of Decision Trees: Example



Decision Tree, T_L



Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

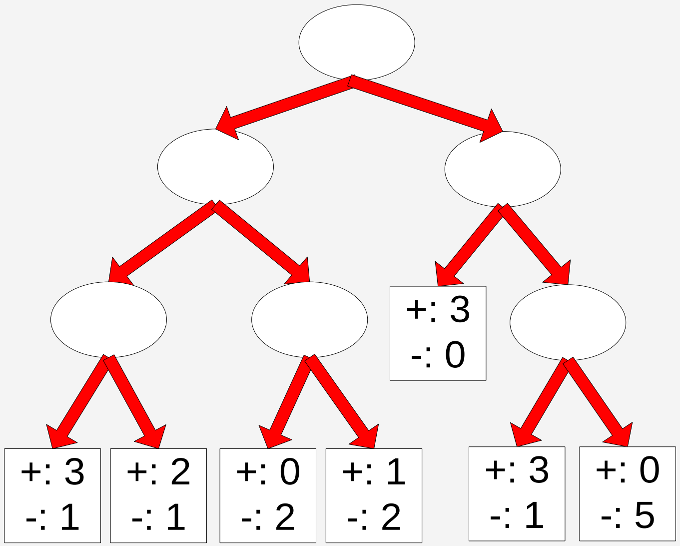
$\Omega = 1$

$e_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

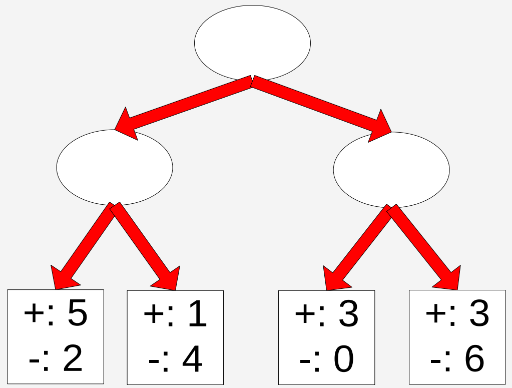
$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

Estimating the Complexity of Decision Trees

- Resubstitution Estimate:
  - Using training error as an optimistic estimate of generalization error
  - Referred to as optimistic error estimate



Decision Tree, T_L



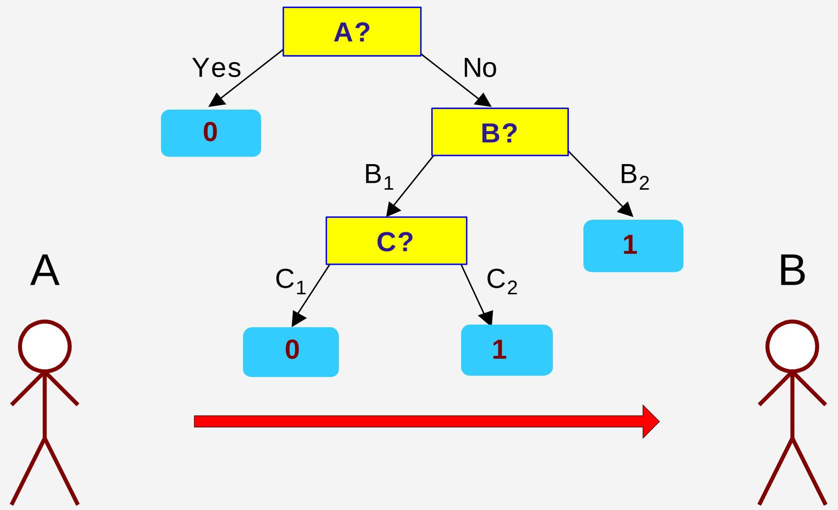
Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

Minimum Description Length (MDL)

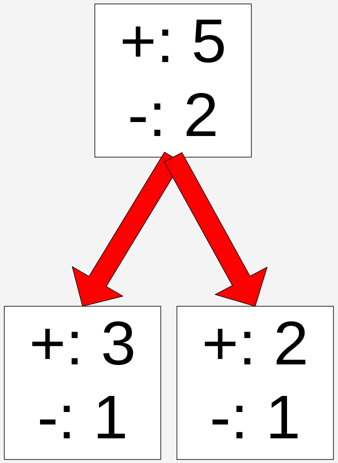
X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

- Cost(Model,Data) = Cost(Data|Model) + α x 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 Statistical Bounds



$$e'(N,e,\alpha) = \frac{e + \frac{z_{\alpha/2}^2}{2N} + z_{\alpha/2}\sqrt{\frac{e(1-e)}{N} + \frac{z_{\alpha/2}^2}{4N^2}}}{1 + \frac{z_{\alpha/2}^2}{N}}$$

Before splitting: e = 2/7, e'(7, 2/7, 0.25) = 0.503

e'(T) = 7 × 0.503 = 3.521

After splitting:

e(T_L) = 1/4, e'(4, 1/4, 0.25) = 0.537

e(T_R) = 1/3, e'(3, 1/3, 0.25) = 0.650

e'(T) = 4 × 0.537 + 3 × 0.650 = 4.098

Therefore, do not split

Model Selection for Decision Trees

- 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).
    - Stop if estimated generalization error falls below certain threshold

Model Selection for Decision Trees

- Post-pruning
  - Grow decision tree to its entirety
  - Subtree replacement
    - Trim the nodes of the decision tree in a bottom-up fashion
    - If generalization error improves after trimming, replace sub-tree by a leaf node
    - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Subtree raising
    - Replace subtree with most frequently used branch

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

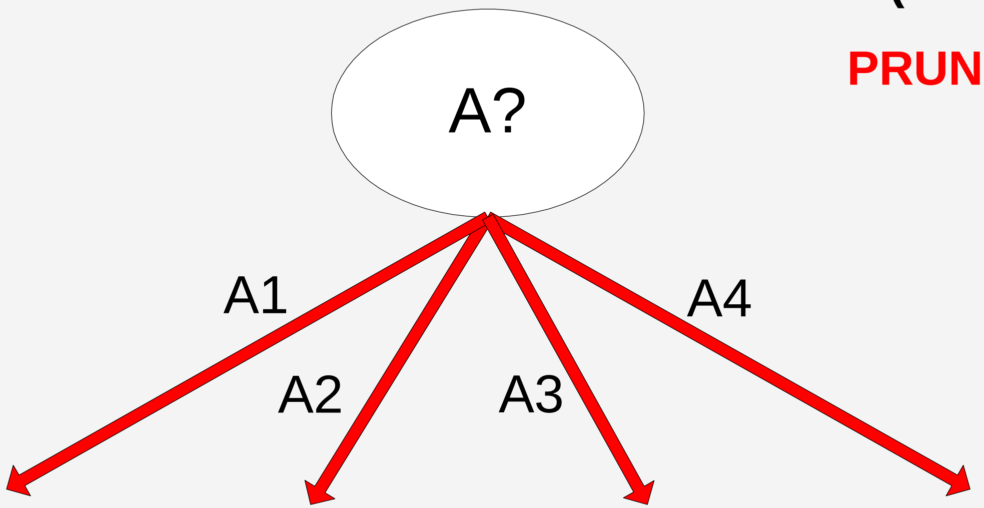
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

$= (9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Examples of Post-pruning

<u>Decision Tree:</u>

depth = 1 :
| breadth > 7 : **class 1**
| breadth <= 7 :
| | breadth <= 3 :
| | | ImagePages > 0.375 : **class 0**
| | | ImagePages <= 0.375 :
| | | | totalPages <= 6 : **class 1**
| | | | totalPages > 6 :
| | | | | breadth <= 1 : **class 1**
| | | | | breadth > 1 : **class 0**
| | width > 3 :
| | | MultiIP = 0:
| | | | ImagePages <= 0.1333 : **class 1**
| | | | ImagePages > 0.1333 :
| | | | | breadth <= 6 : **class 0**
| | | | | breadth > 6 : **class 1**
| | | MultiIP = 1:
| | | | TotalTime <= 361 : **class 0**
| | | | TotalTime > 361 : **class 1**
depth > 1 :
| MultiAgent = 0:
| | depth > 2 : **class 0**
| | depth <= 2 :
| | | MultiIP = 1: **class 0**
| | | MultiIP = 0:
| | | | breadth <= 6 : **class 0**
| | | | breadth > 6 :
| | | | | RepeatedAccess <= 0.0322 : **class 0**
| | | | | RepeatedAccess > 0.0322 : **class 1**
| MultiAgent = 1:
| | totalPages <= 81 : **class 0**
| | totalPages > 81 : **class 1**

Subtree Raising

Subtree Replacement

<u>Simplified Decision Tree:</u>

depth = 1 :
| ImagePages <= 0.1333 : **class 1**
| ImagePages > 0.1333 :
| | breadth <= 6 : **class 0**
| | breadth > 6 : **class 1**
depth > 1 :
| MultiAgent = 0: **class 0**
| MultiAgent = 1:
| | totalPages <= 81 : **class 0**
| | totalPages > 81 : **class 1**

Model Evaluation

- Purpose:
  - To estimate performance of classifier on previously unseen data (test set)
- Holdout
  - Reserve k% for training and (100-k)% for testing
  - Random subsampling: repeated holdout
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n

Cross-validation Example

- 3-fold cross-validation