Data Mining

Model Overfitting

Introduction to Data Mining, 2nd Edition

by

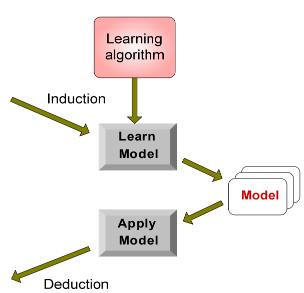
Tan, Steinbach, Karpatne, Kumar

Classification Errors

- **Training 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

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

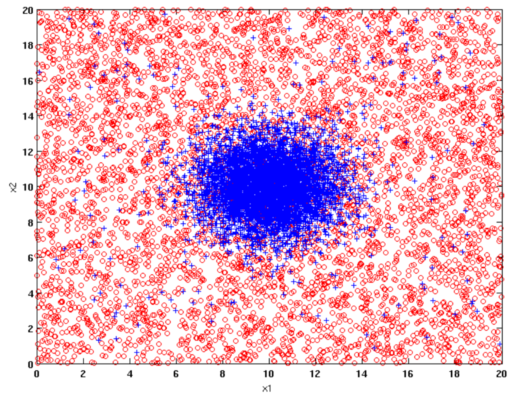
Training Set



Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

Example Data Set



Two class problem:

+ : 5400 instances

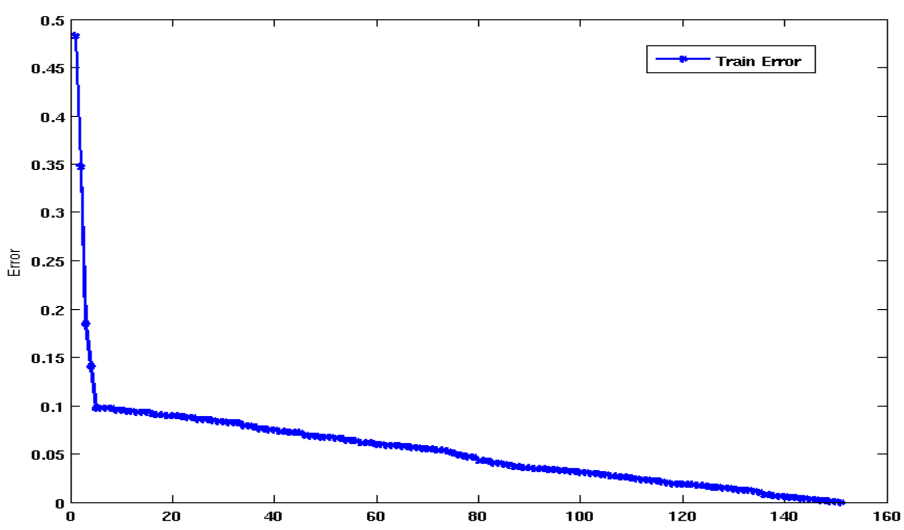
- 5000 instances generated from a Gaussian centered at (10,10)
- 400 noisy instances added

o : 5400 instances

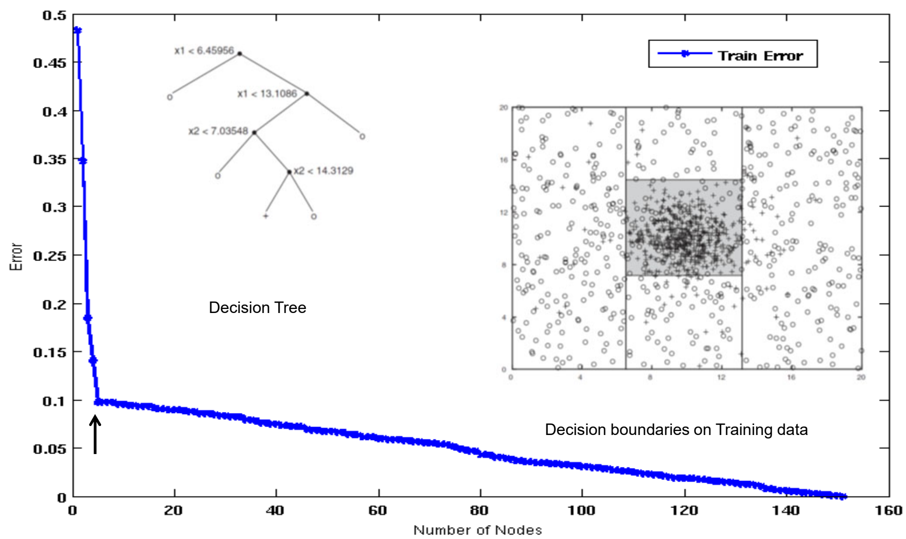
- Generated from a uniform distribution

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

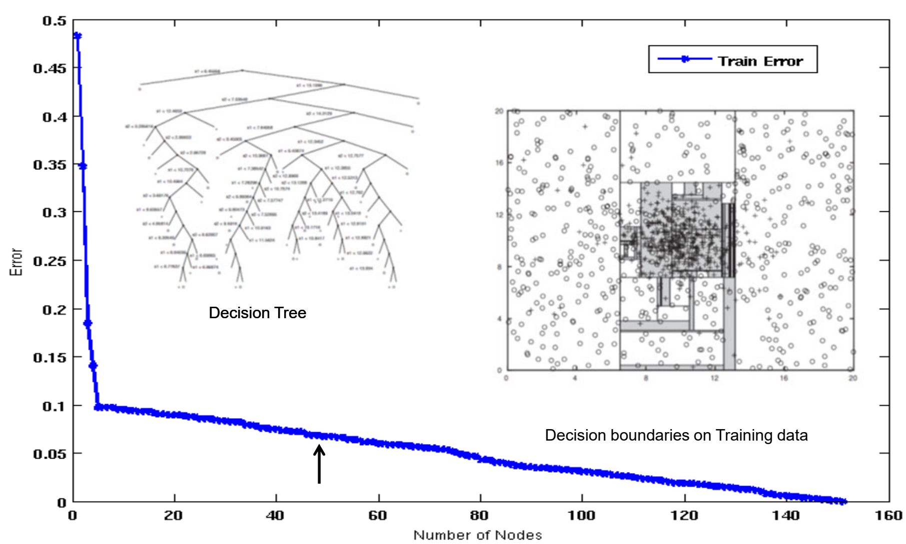
Increasing number of nodes in Decision Trees



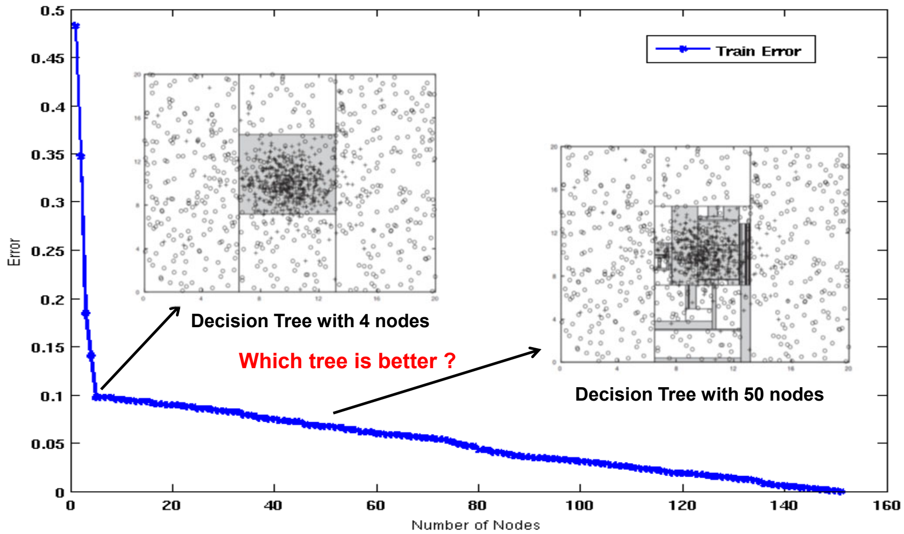
Decision Tree with 4 nodes



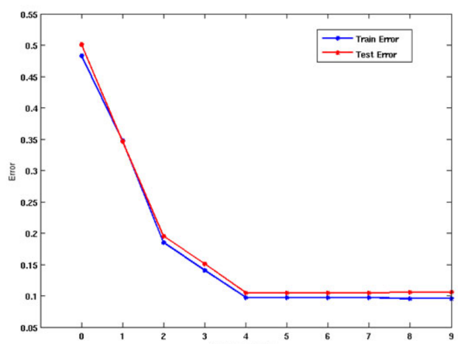
Decision Tree with 50 nodes

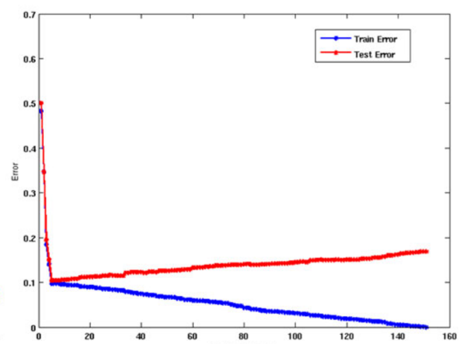


Which tree is better?



Model Underfitting and Overfitting



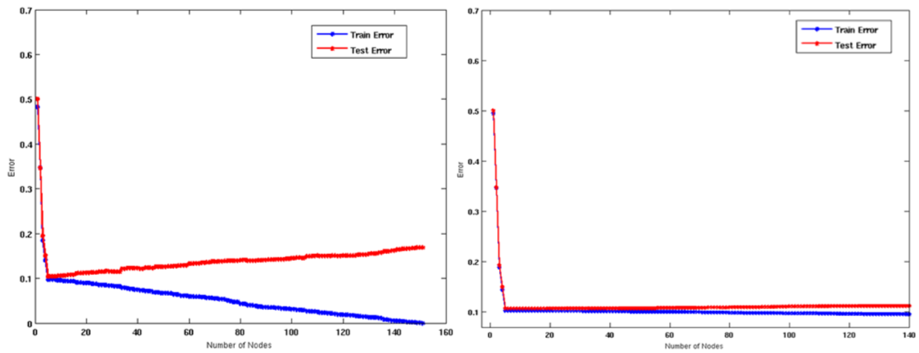


- As the model becomes more and more complex, test errors can start increasing even though training error may be decreasing

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

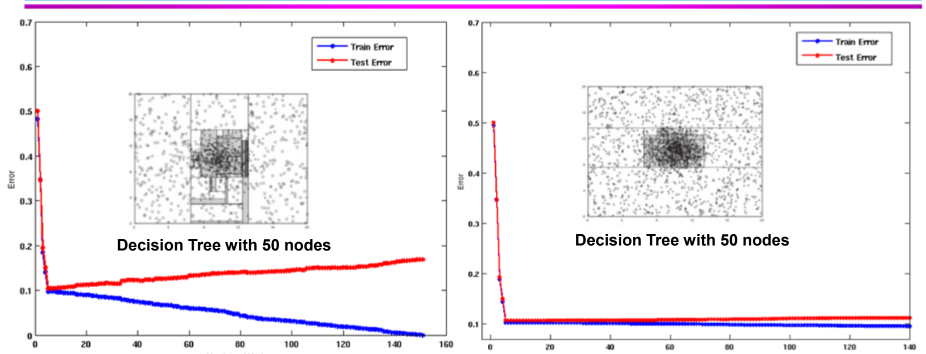
Model Overfitting – Impact of Training Data Size



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 – Impact of Training Data Size



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

Reasons for Model Overfitting

- Not enough training data

- 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

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

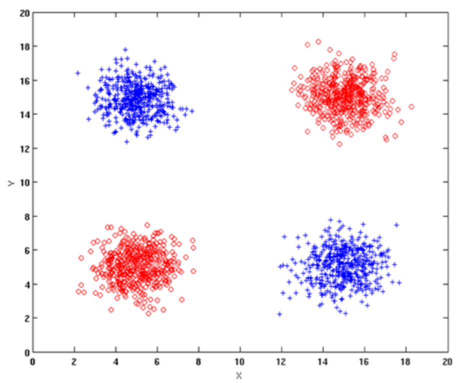
Effect of Multiple Comparison Procedure

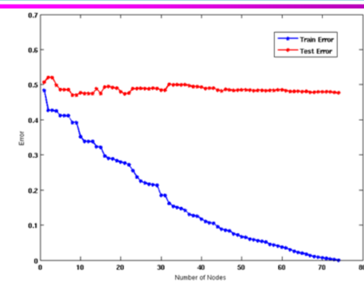
- Many algorithms employ the following greedy strategy:
  - Initial model: M
  - Alternative model: M' = M $\cup$ γ,
    where γ is a component to be added to the model (e.g., a test condition of a decision tree)
  - Keep M' if improvement, Δ(M,M') > α

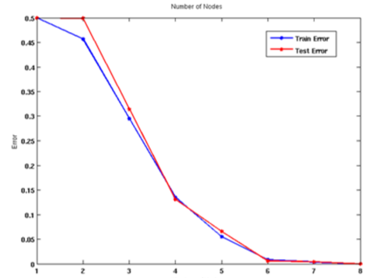
- Often times, γ is chosen from a set of alternative components, $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$

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

Effect of Multiple Comparison - Example







Use additional 100 noisy variables generated from a uniform distribution along with X and Y as attributes.

Use 30% of the data for training and 70% of the data for testing

Using only X and Y as attributes

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

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

Model Selection: Using Validation Set

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

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

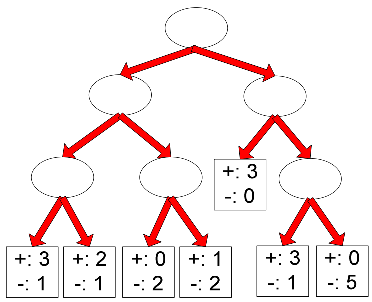
Estimating the Complexity of Decision Trees

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

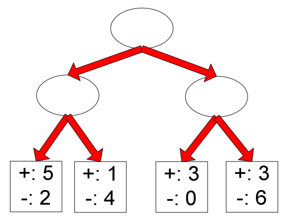
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

  - 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

$e(T_L) = 4/24$

$e(T_R) = 6/24$

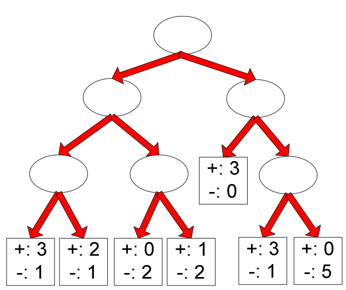
$\Omega = 1$

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

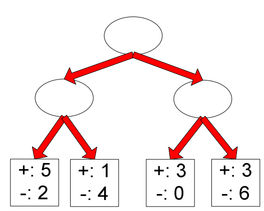
$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

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



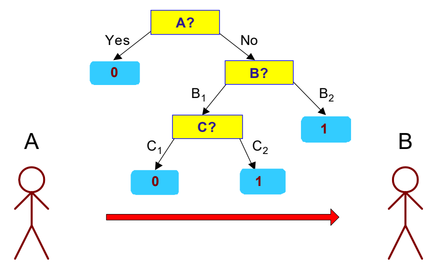
Decision Tree, T_R

$e(T_L) = 4/24$

$e(T_R) = 6/24$

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.

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

Example of Post-Pruning

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

Training Error (Before splitting) = 10/30

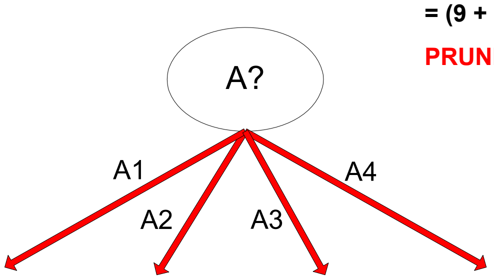
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

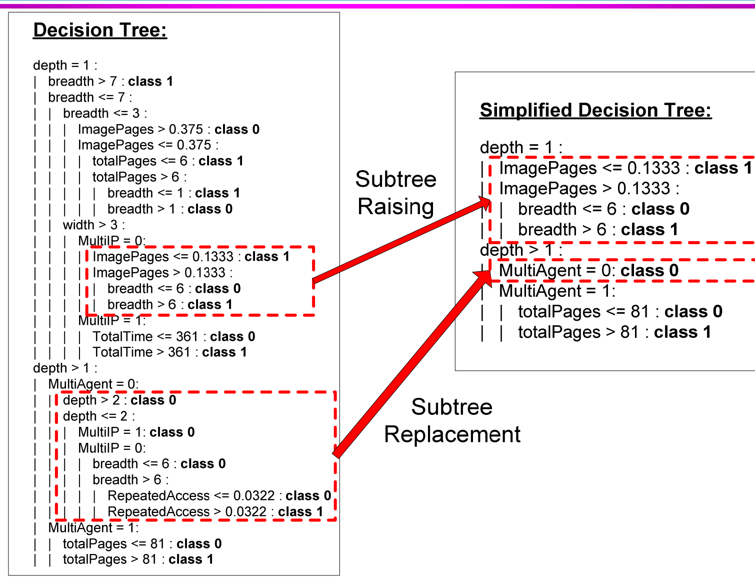
= $(9 + 4 \times 0.5)/30 = 11/30$

PRUNE!



Class = Yes	8	Class = Yes	3	Class = Yes	4	Class = Yes	5
Class = No	4	Class = No	4	Class = No	1	Class = No	1

Examples of Post-pruning

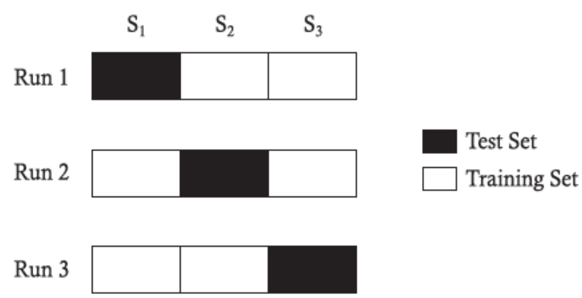


Model Evaluation

- 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



Variations on Cross-validation

- Repeated cross-validation
  – Perform cross-validation a number of times
  – Gives an estimate of the variance of the generalization error
- Stratified cross-validation
  – Guarantee the same percentage of class labels in training and test
  – Important when classes are imbalanced and the sample is small
- Use nested cross-validation approach for model selection and evaluation