# Introduction to Machine Learning

Ranga Raju Vatsavai, Ph.D.
Chancellors Faculty Excellence Associate Professor in Geospatial Analytics
Department of Computer Science, North Carolina State University (NCSU)

Feb. 25-27, 2019

## Characteristics of Decision Tree Induction

- Non-parametric classification
  - No prior assumptions about how data is generated (prob. dist.)
- Finding optimal tree is NP-complete
  - Heuristic: greedy, top-down, recursive-partitioning
- Robust to noise: built-in techniques to avoid **overfitting**
- *Redundant* attributes: does not adversely affect the accuracy, but *irrelevant* attributes may lead to larger trees (than necessary) – feature selection techniques during preprocessing may help to improve accuracy
- Data fragmentation: number of samples get reduced as you traversed down the trees. It may happen that number of samples at leaf node may be insufficient to make statistically significant decisions about classes.
- Subtree replication may happen: (i) single attribute test condition, and (ii) due to divide-and-conquer strategy same test can be applied to different parts of the attribute space
- Single attribute at a time – problem for continuous attributes – *solution*: oblique decision trees
- Choice of impurity measures has little effect on the performance of decision tree induction
  - Gini, Entropy, Classification Error
- Strategy to prune tree has greater impact on final tree than choice of impurity measure

## Decision Boundaries Induced by DT's

- Tree growing can be viewed as partitioning the attribute space into disjoint regions

## Classification Errors

- Training errors (aka resubstitution or 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 with 50 nodes

# Increasing number of nodes in Decision Trees

# 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

# Reasons for Model Overfitting

- Presence of Noise
- Lack of Representative Samples

## Lack of Representative Samples

**Table 4.5.** An example training set for classifying mammals.

| Name | Body Temperature | Gives Birth | Four-legged | Hibernates | Class Label |
|---|---|---|---|---|---|
| salamander | cold-blooded | no | yes | yes | no |
| guppy | cold-blooded | yes | no | no | no |
| eagle | warm-blooded | no | no | no | no |
| poorwill | warm-blooded | no | no | yes | no |
| platypus | warm-blooded | no | yes | yes | yes |

Humans | warm-blooded | yes | no |no | yes

**Figure 4.26.** Decision tree induced from the data set shown in Table 4.5.

## 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
  - How do we determine the right model complexity

## Estimating Generalization Errors

- Resubstitution Estimate
- Incorporating Model Complexity
- Use Validation Set

## Resubstitution Estimate

- Using training error as an optimistic estimate of generalization error

$e(T_L) = 4/24$

$e(T_R) = 6/24$

# 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

# Pessimistic Estimate

- Given a decision tree node *t*
  - $n(t)$: number of training records classified by *t*
  - $e(t)$: number of errors at node *t*
  - Training error of tree *T* with *k* leaf nodes:

$$e'(T) = \frac{\sum_i \left[ e(t_i) + \Omega(t_i) \right]}{\sum_i n(t_i)} = \frac{e(T) + \Omega(T)}{N}$$

- $\Omega$: is the cost of adding a leaf node
- N: total number of training records

# Pessimistic Estimate

$e(T_L) = 4/24$

$e(T_R) = 6/24$

$\Omega = 1$

$e'(T_L) = (4 + 7 \times 1)/24 = 11/24 = 0.458$

$e'(T_R) = (6 + 4 \times 1)/24 = 10/24 =$ **0.417**

What if $\Omega = 0.5$

# 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) + 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.

## 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

## Handling Overfitting in Decision Tree

- 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 $\chi^2$ 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

## Handling Overfitting in Decision Tree

- 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 × 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 |

## Subtree Replacement Example

- Bottom-up
- Consider replacing a tree only after considering all its subtrees

© J. Furnkranz

## Subtree Replacement Example

© J. Furnkranz

- Delete node B
- Redistribute instances of leaves 4 and 5 into C

## Evaluating Performance of Classifier

- Model Selection
  - Performed during model building
  - Purpose is to ensure that model is not overly complex (to avoid overfitting)
  - Need to estimate generalization error
- Model Evaluation
  - Performed after model has been constructed
  - Purpose is to estimate performance of classifier on previously unseen data (e.g., test set)

## Methods for Classifier Evaluation

- 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
- Bootstrap
  - Sampling with replacement
  - .632 bootstrap: $acc_{boot} = \frac{1}{b}\sum_{i=1}^{b}(0.632 \times acc_i + 0.368 \times acc_s)$

NC STATE UNIVERSITY

Any Questions?