## CS522 Advanced Database Systems

Classification: Basic Concepts and Decision Trees

Chengyu Sun
California State University, Los Angeles

## A Classification Problem

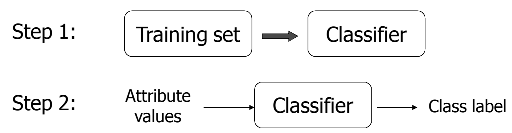
- Is a loan to a person who is 45 years old, divorced, renting an apartment, with two kids and annual income of 100K high risk or low risk?

| **Age** | **Home Owner** | **Marital Status** | **# of Kids** | **Annual Income** | **Risk** |
|---|---|---|---|---|---|
| 45 | No | Divorced | 2 | 100K | ? |

## Terminology and Concepts ...

- Record (or tuple)
  - Attributes
    - E.g. age, marital status, # of kids, owns home or not, credit score ...
  - Class label
    - E.g. high risk, low risk ...
- Classification: predict the class label with given attribute values

## ... Terminology and Concepts




- Classifier (or model)
- Training set: records with known class labels that are used to construct (i.e. *train*) the classifier

## Classification vs. Regression

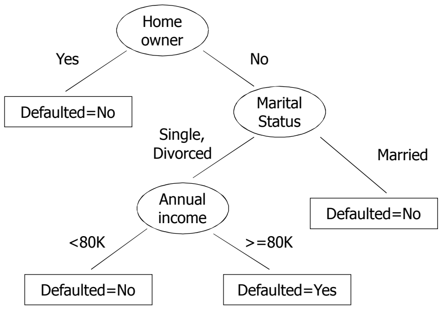
- Classification predicts categorical attribute values
- Regression predicts *continuous* numerical attribute values

| SID | HW1 | HW2 | HW3 | Final | Pass/Fail |
|---|---|---|---|---|---|
| 1 | 40 | 60 | 70 | 95 | Passed |
| 2 | 10 | 15 | 11 | 65 | Failed |
| 3 | 30 | 45 | 40 | 75 | Passed |
| 4 | 35 | 50 | 35 | ? | ? |

## A Training Set

| **TID** | **Home Owner** | **Marital Status** | **Annual Income** | **Defaulted Borrower** |
|---|---|---|---|---|
| 1 | Yes | Single | 125K | No |
| 2 | No | Married | 100K | No |
| 3 | No | Single | 70K | No |
| 4 | Yes | Married | 120K | No |
| 5 | No | Divorced | 95K | Yes |
| 6 | No | Married | 60K | No |
| 7 | Yes | Divorced | 220K | No |
| 8 | No | Single | 85K | Yes |
| 9 | No | Married | 75K | No |
| 10 | No | Single | 90K | Yes |

## A Decision Tree



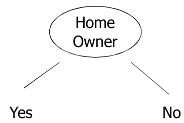

## Decision Tree Induction

- Let D be the set of training record associated with current node
  - If all record in D belong to the same class C, current node is a leaf node and is labeled as C.
  - If D contains records that belong to more than one class, *select an attribute to split* D *into subsets*, and create a child node for each subset. Apply the algorithm recursively on each child node.

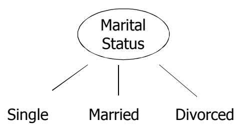
## Terminating Conditions

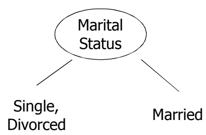
- All records in D belong to the same class
- No more attribute to split
  - *Class label??*
- No records associated with the node
  - *Class label??*

## Split on Binary Attributes

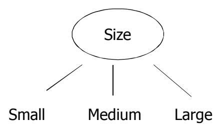


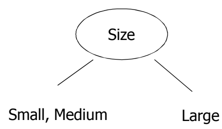

## Split on Nominal Attributes



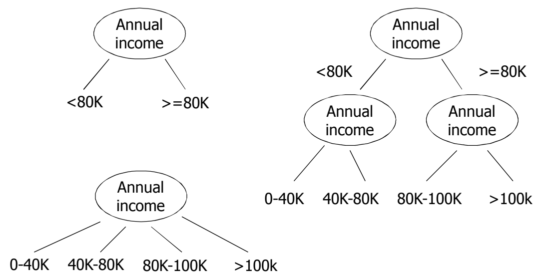




## Split on Ordinal Attributes



## Split on Continuous Attributes




## Discretization of Numerical Data

- Number of partitions is known
  - Equi-width
  - Equi-depth
- Number of partitions is unknown
  - Binary split and recursive binary split
    - Determine the best split point
  - Intuitive partitioning

## Equi-Width vs. Equi-Depth

- Dataset: 10,21,23,25,26,29,30,35,39
- # of Partitions: 3
- Equi-width: ??
- Equi-depth: ??

## Intuitive Partitioning

- The 3-4-5 Rule
  - 3, 6, 7, or 9 distinct values at the most significant digit ➔ 3 equi-width intervals (2-3-2 for 7)
  - 2,4,8 ➔ 4 equi-width intervals
  - 1,5,10 ➔ 5 equi-width intervals
- Example: 60 70 75 85 90 95 100 120 125 220
  - Intervals??

## Splitting Attribute Selection

- After a split, ideally each subset would "pure", i.e. contains only one class of records

| **Gender** | **Age** | **Preferred color** |
|---|---|---|
| female | 20 | pink |
| male | 20 | black |
| female | 15 | pink |
| male | 15 | black |

## Attribute Selection Measures

- Entropy (Information Gain)
- Gini index
- Gain Ratio

## Entropy

$$Entropy(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- $p_i$ is the fraction of records in D that belongs to class $C_i$
- m is the number of classes in D

## Entropy Example

- Preferred color
  - 2 black and 2 pink??
  - 3 black and 1 pink??
  - 4 black??

## Information Gain

- Suppose D is split into v subsets on attribute A

$$Gain(A) = Entropy(D) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Entropy(D_j)$$

## Information Gain Example

- Preferred color
  - Gain(Gender)??
  - Gain(Age)??

## Split Information

- Information gain favors attributes with lots of distinct values
- *Split information* can be used to "normalized" information gain

$$SplitInfo(A) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

## Gain Ratio

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

## Gini Index

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2$$

- Used in the CART algorithm for *binary split*

## Gini Index Example

- Preferred color
  - 2 black and 2 pink??
  - 3 black and 1 pink??
  - 4 black??
  - Split on gender??
  - Split on age??

## Decision Tree Induction Example

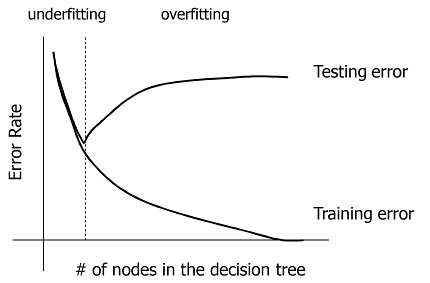
| A1 | A2 | A3 | C |
|---|---|---|---|
| Y | L | 20 | C1 |
| Y | S | 9 | C2 |
| N | S | 11 | C2 |
| Y | M | 14 | C1 |
| N | L | 14 | C1 |
| Y | S | 15 | C1 |

*How do we make the first split??*

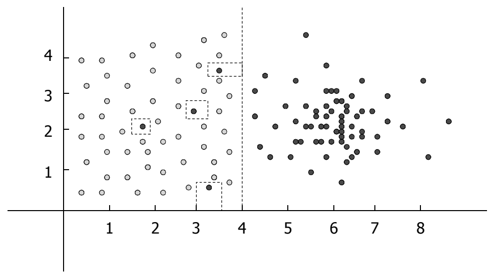
## Training Error and Testing Error

- Training error
  - Misclassification of training records
- Testing (Generalization) error
  - Misclassification of testing records

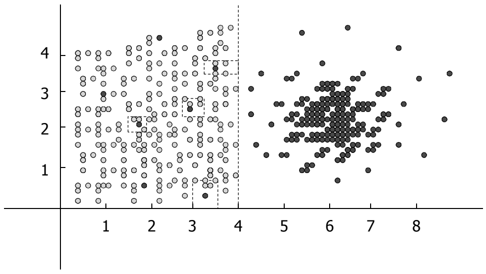
## Model Overfitting and Underfitting




## Overfitting Due to Outliers/Noise ...



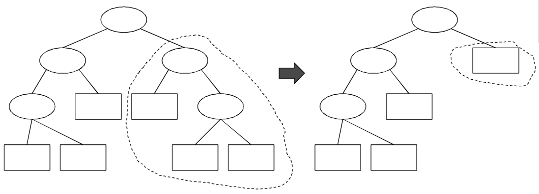
## ... Overfitting Due to Outliers/Noise




## Occam's Razor

- A.K.A. Principle of Parsimony
- Given two models with the same generalization errors, the simpler model is preferred over the more complex model

## Tree Pruning



- Replace a subtree with a leaf node
- The class label of the leaf is the majority class label of the records associated with the subtree

## Prepruning

- Prune during decision tree construction
  - Number of records < threshold
  - "Purity gain" < threshold

## Postpruning

- Buttom-up pruning of a fully constructed tree
  - Replace a subtree with a leaf node if it reduces testing error
    - *How do we know whether it reduces testing error or not??*
  - Pruning based on Minimum Description Length (MDL)

## Estimate Testing Errors ...

- Use a *pruning/validation set*
  - Usually 1/3 of the original training set
  - Less records for training

## ... Estimate Testing Errors

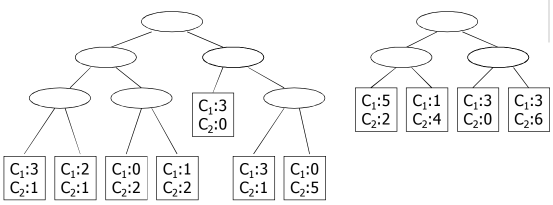
- Optimistic error estimation
  - The training set is a good representation of the overall data (optimistic!), so the training error is the testing error
- Pessimistic error estimation
  - Training error + penalty term for model complexity

## Pessimistic Error Estimation

$$e_g(T) = \frac{\sum_i [e(t_i) + \Omega(t_i)]}{\sum_i n(t_i)} = \frac{e(T) + \Omega(T)}{N}$$

- T – A decision tree
- $n(t_i)$ – # of training records at leaf node $t_i$
- $e(t_i)$ – # of misclassified training records at $t_i$
- $\Omega(t_i)$ – Penalty term associated with $t_i$

## Example of Pessimistic Error Estimation

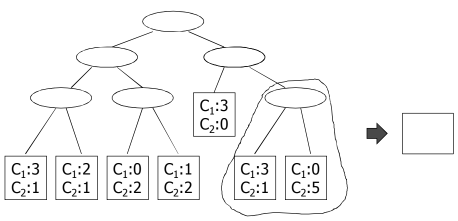



- $e_g(T)$ with $\Omega(t_i)$=0.5?? $\Omega(t_i)$=1??

## About $\Omega(t_i)$

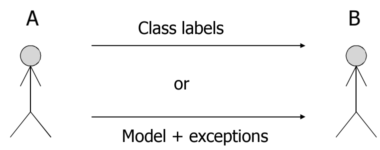
- The right value of $\Omega(t_i)$ depends on the branching factor of the decision tree
- For example, for a binary decision tree, $\Omega(t_i) < 0.5$ or $\Omega(t_i) > 0.5$??

## Pruning with Estimated Testing Error




- With optimistic error estimation??
- With pessimistic error estimation??

## Minimum Description Length (MDL)

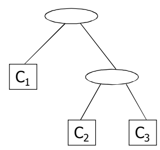


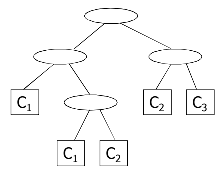

- The best model is the one that minimizes the number of bits to encode both the model and the exceptions to the model

## MDL Example ...

- $n$ records
- $m$ binary attributes
- $k$ classes
- Cost(Internal Node) = $\log_2 m$
- Cost(Leaf node) = $\log_2 k$
- Cost(Error) = $\log_2 n$
- Cost = Cost(All Nodes)+Cost(All Errors)

## ... MDL Example



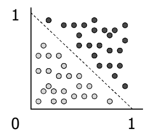

- 128 records, 8 binary attributes and 3 classes
- Left tree has 7 errors and right tree has 4 errors

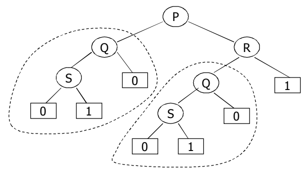
## About Decision Tree Classification ...

- Inexpensive to construct
- Extremely fast at classifying unknown records
- Easy to interpret for small-sized trees
- Accuracy is comparable to other classification techniques for many simple data sets

## ... About Decision Tree Classification

- Data fragmentation
- Tree replication
- Finding an optimal decision tree is NP-hard
- Limitation on expressiveness




## Readings

- Texbook Chapter 6.1, 6.2, and 6.3