Cse352
Artificial Intelligence
Classification Learning
Short Review

Professor Anita Wasilewska
Computer Science Department
Stony Brook University
Preprocessing stage

• **Preprocessing:**
  - includes all the operations that have to be performed before a data mining, learning algorithm is applied

• **Data in the real world is dirty:** incomplete, noisy and inconsistent.

• **Quality decisions** must be based on quality Data.
Preprocessing stage

• Data cleaning
  – Fill in missing values, smooth noisy data (binning, clustering, regression), identify or remove outliers, and resolve inconsistencies

• Data integration
  – Integration of multiple databases, data cubes, or files
Preprocessing stage

- **Data transformation**

- **Normalization** and aggregation
- **Data reduction** and attribute selection
- Obtains reduced presentation in volume but produces the same or similar analytical results (stratified sampling, PCA, cluster)

- **Data discretization**
- Part of data reduction but reduces the number of values of the attributes by dividing the range of attributes into intervals (segmentation by natural partition, hierarchy generation)
• DM, Learning proper is a step in the
• DM, Learning process in which algorithms are applied to obtain patterns in data.

• It can be re-iterated- and usually is
Descriptive/Non-Descriptive Models

• **Statistical and Descriptive**

• **Statistical models** use historical data to predict some unknown or missing numerical values

• **Descriptive models** aim to find patterns in the data that provide some information about what the data contains

• In case of **Classification** they often present the knowledge as a set of rules of the form **IF.... THEN...**
Classifications Models

- **Descriptive:** Decision Trees, Rough Sets, Classification by Association

- **Statistical:** Neural Networks, Bayesian Networks, Cluster, Outlier analysis, Trend and evolution analysis

- **Optimization method:** Genetic Algorithms – can be descriptive
Classification

• **Classification:**
• Finding models (rules) that describe (characterize) or/and distinguish (discriminate) classes or concepts for future prediction

• **Classification Data Format:**
• a data table with key attribute removed.
• Special attribute, called a class attribute must be distinguished.
• The values: $c_1, c_2, \ldots, c_n$ of the class attribute $C$ are called class labels
• The class label attributes are discrete valued and unordered.
Classification

• Goal:

• **FIND** a minimal set of **characteristic and/or discriminant rules**, or **other descriptions** of a class $C$, or all classes, or some classes

• We also want the found rules to involve as few attributes as it is possible
Classification

- **Stage 1:** build the basic patterns structure-training
- **Stage 2:** optimize parameter settings; can use (N:N) re-substitution- **parameter tuning**
- Re-substitution error rate = training data error rate
- **Stage 3:** use **test data** to compute- **predictive accuracy/error rate** - **testing**
Decision Tree

- **DECISION TREE**
- A flow-chart-like tree structure;
- **Internal node** denotes an **attribute**;
- **Branch** represents the **values** of the node attribute;
- **Leaf nodes** represent **class labels**
The basic DT algorithm for decision tree construction is a greedy algorithm that constructs decision trees in a top-down recursive divide-and-conquer manner.

Tree STARTS as a single node representing all training dataset (data table with records called samples)

IF the samples (records in the data table) are all in the same class, THEN the node becomes a leaf and is labeled with that class

The algorithm uses the same process recursively to form a decision tree at each partition
DT Basic Algorithm

• The recursive partitioning **STOPS** only when any one of the following conditions is TRUE

• **1. All records** (samples) for the given node belong to the same class

• **2. There are no remaining attributes** on which the samples (records in the data table) may be further partitioned – a LEAF is created with **majority vote** for training sample

• **3. There is no records** (samples) left – a LEAF is created with **majority vote** for training sample

• **Majority voting** involves converting node N into a leaf and labeling it with **the most common class in D** which is a set of training tuples and their associated class labels
Attribute Selection Measures

- **Some Heuristics:**
- Some **Decision Tree** Attribute Selection Measures are:
  - **Information Gain, Gini Index**

- We use them for selecting the attribute that “best” discriminates the given tuples according to class
Neural Networks

- **Neural Network** is a set of connected **INPUT/OUTPUT UNITS**, where each connection has a **WEIGHT** associated with it.
- **Neural Network learns** by adjusting the weights so as to be able to **correctly classify** the training data and hence, **after testing phase**, to **classify unknown data**.
- **Neural Network** needs long time for training. Determining **network topology** is difficult.
- Choosing single learning rate impossible (train with subset).
- **Neural Network** has a **high tolerance** to noisy and incomplete data.
- **NN is generally better** with larger number of hidden units.
Neural Networks

- The inputs to the network correspond to the attributes and their values for each training tuple.
- Inputs are fed simultaneously into the units making up the input layer.
- Inputs are then weighted and fed simultaneously to a hidden layer.
- The number of hidden layers is arbitrary, although often only one or two.
- The weighted outputs of the last hidden layer are input to units making up the output layer, which emits the network's prediction.
Neural Networks

• For each training sample, the weights are first set random then they are modified as to minimize the mean squared error between the network’s classification (prediction) and actual classification.

• Backpropagation Algorithm:
  • STEP ONE: initialize the weights and biases
  • STEP TWO: feed the training sample
  • STEP THREE: propagate the inputs forward
  • STEP FOUR: backpropagate the error
  • STEP FIVE: backpropagate the weights, biases
  • STEP SIX: repeat and apply Terminating Conditions
Backpropagation Formulas

Output vector

Output nodes

Hidden nodes

Input nodes

Input vector: \( x_i \)

\[
Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}
\]

\[
\theta_j = \theta_j + (l) Err_j
\]

\[
w_{ij} = w_{ij} + (l) Err_j O_i
\]

\[
Err_j = O_j (1 - O_j) (T_j - O_j)
\]

\[
O_j = \frac{1}{1 + e^{-I_j}}
\]

\[
I_j = \sum_i w_{ij} O_i + \theta_j
\]
Back propagation

• **Terminating Conditions:**

• **Process Stops** when:

• All *wij* in the previous epoch are below some threshold

• The percentage of samples *misclassified* in the previous epoch is below some threshold

• a pre-specified number of epochs has expired
Building a classifier

- **Building a classifier** consists of two phases: *training* and *testing*.
- We use the *training data* set to *create patterns*: rules, trees, or to *train* a Neural or Bayesian network.
- We evaluate created patterns with the use of *test data*.
- We *terminate* the process of building a classifier if it has been *trained* and *tested* and the *predictive accuracy* is on an acceptable level.
- **CLASSIFIER** is a *final product* of the process.

- **PREDICTIVE ACCURACY** of a classifier is a percentage of well classified data in the test data set.
Training and Testing

• The **main methods** of **predictive accuracy evaluations** are:
  • Re-substitution \((N ; N)\)
  • Holdout \((2N/3 ; N/3)\)
  • k-fold cross-validation \((N- N/k ; N/k)\)
  • Leave-one-out \((N-1 ; 1)\)