Classification fundamentals

Elena Baralis
Politecnico di Torino

Classification

Objectives
- prediction of a class label
- definition of an interpretable model of a given phenomenon

Approaches
- decision trees
- bayesian classification
- classification rules
- neural networks
- k-nearest neighbours
- SVM

Requirements
- accuracy
- interpretability
- scalability
- noise and outlier management

Applications
- detection of customer propension to leave a company (churn or attrition)
- fraud detection
- classification of different pathology types
- ...

Classification: definition

Given
- a collection of class labels
- a collection of data objects labelled with a class label

Find a descriptive profile of each class, which will allow the assignment of unlabeled objects to the appropriate class
Definitions

- **Training set**
  - Collection of labeled data objects used to learn the classification model

- **Test set**
  - Collection of labeled data objects used to validate the classification model

Classification techniques

- **Decision trees**
- **Classification rules**
- **Association rules**
- **Neural Networks**
- **Naïve Bayes and Bayesian Networks**
- **k-Nearest Neighbours (k-NN)**
- **Support Vector Machines (SVM)**
- ...

Evaluation of classification techniques

- **Accuracy**
  - Quality of the prediction

- **Efficiency**
  - Model building time
  - Classification time

- **Scalability**
  - Training set size
  - Attribute number

- **Robustness**
  - Noise, missing data

- **Interpretability**
  - Model interpretability
  - Model compactness

Example of decision tree

<table>
<thead>
<tr>
<th>Tid</th>
<th>Refund</th>
<th>Single</th>
<th>Married</th>
<th>Divorced</th>
<th>Taxable Income</th>
<th>Cheat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Single</td>
<td>Married</td>
<td>No</td>
<td>125K</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>Married</td>
<td>Divorced</td>
<td>No</td>
<td>100K</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Single</td>
<td>Single</td>
<td>No</td>
<td>70K</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Married</td>
<td>Married</td>
<td>No</td>
<td>120K</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Divorced</td>
<td>Divorced</td>
<td>Yes</td>
<td>95K</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>Married</td>
<td>Married</td>
<td>No</td>
<td>60K</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>Divorced</td>
<td>Married</td>
<td>No</td>
<td>220K</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>Single</td>
<td>Single</td>
<td>Yes</td>
<td>85K</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>No</td>
<td>Married</td>
<td>Married</td>
<td>No</td>
<td>75K</td>
<td>No</td>
</tr>
<tr>
<td>10</td>
<td>No</td>
<td>Single</td>
<td>Single</td>
<td>Yes</td>
<td>90K</td>
<td>Yes</td>
</tr>
</tbody>
</table>

There could be more than one tree that fits the same data!
Apply Model to Test Data

Start from the root of the tree.

Refund: Yes

MarSt: No

TaxInc: < 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: Yes

Refund: No

MarSt: Single, Divorced

TaxInc: < 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: > 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: < 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: > 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: < 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: > 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: < 80K

Assign Cheat to "No"

Refund: No

MarSt: Married

TaxInc: > 80K

Assign Cheat to "No"
Data mining: classification

**Decision tree induction**

- Many algorithms to build a decision tree
  - Hunt’s Algorithm (one of the earliest)
  - CART
  - ID3, C4.5, C5.0
  - SLIQ, SPRINT

**Hunt’s algorithm**

```
          Best A
          /   \
         /     \
  Don’t Cheat  Yes  No
  /   \
 /     \
Single       Married
/   \
|     |
|     |
< 80K ≥ 80K
```

**Decision tree induction**

- Adopts a greedy strategy
  - "Best" attribute for the split is selected locally at each step
  - not a global optimum

**Issues**

- Structure of test condition
  - Binary split versus multiway split
- Selection of the best attribute for the split
- Stopping condition for the algorithm

**Structure of test condition**

- Depends on attribute type
  - nominal
  - ordinal
  - continuous
- Depends on number of outgoing edges
  - 2-way split
  - multi-way split

**Splitting on nominal attributes**

- Multi-way split
  - use as many partitions as distinct values
- Binary split
  - Divides values into two subsets
  - Need to find optimal partitioning
### Data mining: classification

#### Splitting on ordinal attributes
- Multi-way split
  - use as many partitions as distinct values
- Binary split
  - Divides values into two subsets
  - Need to find optimal partitioning

#### Splitting on continuous attributes
- Different techniques
  - Discretization to form an ordinal categorical attribute
    - Static — discretize once at the beginning
    - Dynamic — discretize during tree induction
  - Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering
  - Binary decision $(A < v)$ or $(A \geq v)$
    - consider all possible splits and find the best cut
    - more computationally intensive

#### Selection of the best attribute
- Attributes with *homogeneous* class distribution are preferred
- Need a measure of node impurity

#### Measures of node impurity
- Many different measures available
  - Gini index
  - Entropy
  - Misclassification error
- Different algorithms rely on different measures
How to find the best attribute

Before Splitting:

**Node N1**

- **A**?
  - Yes
  - **Node N10**
  - No
  - **Node N11**

**Node N2**

- **B**?
  - Yes
  - **Node N20**
  - No
  - **Node N21**

**Gain = M0 – M12 vs M0 – M34**

GINI impurity measure

- **Gain for a given node t**
  - \[ GINI(t) = 1 - \sum_{j} p(j \mid t)^2 \]
  - \( p(j \mid t) \) is the relative frequency of class \( j \) at node \( t \)

- **Maximum (1 - 1/n_c) when records are equally distributed among all classes, implying higher impurity degree**
- **Minimum (0.0) when all records belong to one class, implying lower impurity degree**

\[ GINI = 0.500 \]

\[ GINI = 0.371 \]

Computing GINI index: Boolean attribute

- Splits into two partitions
  - larger and purer partitions are sought for

\[ GINI(N1) = 1 - \left( \frac{5}{7} \right)^2 - \left( \frac{2}{7} \right)^2 = 0.408 \]

\[ GINI(N2) = 1 - \left( \frac{1}{5} \right)^2 - \left( \frac{4}{5} \right)^2 = 0.32 \]

Computing GINI index: Categorical attribute

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

**Examples for computing GINI**

**CarType**

- \{Sports, Luxury\}
- \{Family\}

**Gain(split on B)**

\[ \frac{7}{12} \times 0.408 + \frac{5}{12} \times 0.32 = 0.371 \]

Splitting based on GINI

- Used in CART, SLIQ, SPRINT
- When a node \( p \) is split into \( k \) partitions (children), the quality of the split is computed as

\[ GINI_{split} = \frac{1}{n} \sum_{i=1}^{k} n_i GINI(i) \]

Computing GINI index: Categorical attribute

**Multi-way split**

- (find best partition of values)

**Two-way split**

- (find best partition of values)
Computing GINI index: Continuous attribute

- Binary decision on one splitting value
- Number of possible splitting values
- Number of distinct values
- Each splitting value \( v \) has a count matrix
  - class counts in the two partitions
    - \( A < v \)
    - \( A \geq v \)

\[
\text{Sorted Values} \quad \text{Split Positions} \quad \text{Parent} \quad \text{SplitINFO} \quad \text{Left} \quad \text{Right} \\
\begin{array}{ccccc}
\text{Taxable Income} & > 60K & \text{Yes} & \text{No} & \text{No} \\
\text{Categorical A} & Single & 15K & No & No & \text{No} \\
& Married & 15K & No & No & \text{No} \\
& Single & 7K & No & No & \text{No} \\
& Married & 7K & No & No & \text{No} \\
& Single & 5K & No & No & \text{No} \\
& Married & 5K & No & No & \text{No} \\
& Single & 3K & No & No & \text{No} \\
& Married & 3K & No & No & \text{No} \\
& Single & 1K & No & No & \text{No} \\
& Married & 1K & No & No & \text{No} \\
& Single & 0K & No & No & \text{No} \\
& Married & 0K & No & No & \text{No} \\
\end{array}
\]

Examples for computing entropy

\[
\text{Entropy}(t) = -\sum p(j | t) \log p(j | t)
\]

- Maximum (\( \log n_r \)) when records are equally distributed among all classes, implying higher impurity degree
- Minimum (0.0) when all records belong to one class, implying lower impurity degree
- Entropy based computations are similar to GINI index computations

Gain Ratio

\[
\text{GainRATIO}_{\text{INFO}} = \frac{\text{GAIN}}{\text{SplitINFO}}
\]

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain
Classification error impurity measure

- Classification error at a node \( t \)
  \[
  Error(t) = 1 - \max P(i | t)
  \]

  - Measures misclassification error made by a node
    - Maximum \((1 - 1/n_c)\) when records are equally distributed among all classes, implying least interesting information
    - Minimum \((0, 0)\) when all records belong to one class, implying most interesting information

Examples for computing error

\[
Error(t) = 1 - \max P(i | t)
\]

<table>
<thead>
<tr>
<th>Node</th>
<th>( P(C_1) )</th>
<th>( P(C_2) )</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>0</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- \( P(C_1) = 0/6 = 0 \)  \( P(C_2) = 6/6 = 1 \)
- \( P(C_1) = 6/6 = 1 \)  \( P(C_2) = 0/6 = 0 \)

\[
Error = 1 - \max(0, 1) = 1 - 1 = 0
\]

Comparison among splitting criteria

For a 2-class problem

Decision Tree Based Classification

- Advantages
  - 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

- Disadvantages
  - Accuracy may be affected by missing data

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

Practical Issues of Classification

- Underfitting and overfitting
- Missing Values
- Costs of Classification
Underfitting and Overfitting

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

Overfitting due to Noise

- Overfitting due to noise:
- Decision boundary is distorted by noise point.

How to address overfitting

- 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 the 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).

- Post-pruning:
  - Grow decision tree to its entirety.
  - 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.

Handling missing attribute values

- Missing values affect decision tree construction in three different ways:
  - Affects how impurity measures are computed.
  - Affects how to distribute instance with missing value to child nodes.
  - Affects how a test instance with missing value is classified.

Other issues

- Data Fragmentation
- Search Strategy
- Expressiveness
- Tree Replication

Elena Baralis
Politecnico di Torino
Data mining: classification

Data fragmentation
- Number of instances gets smaller as you traverse down the tree
- Number of instances at the leaf nodes could be too small to make any statistically significant decision

Search strategy
- Finding an optimal decision tree is NP-hard
- The algorithm presented so far uses a greedy, top-down, recursive partitioning strategy to induce a reasonable solution
- Other strategies?
  - Bottom-up
  - Bi-directional

Expressiveness
- Decision tree provides expressive representation for learning discrete-valued function
  - But they do not generalize well to certain types of Boolean functions
  - Example: parity function:
    - Class = 1 if there is an even number of Boolean attributes with truth value = True
    - Class = 0 if there is an odd number of Boolean attributes with truth value = True
  - For accurate modeling, must have a complete tree
- Not expressive enough for modeling continuous variables
  - Particularly when test condition involves only a single attribute at-a-time

Decision boundary
- Border line between two neighboring regions of different classes is known as decision boundary
- Decision boundary is parallel to axes because test condition involves a single attribute at-a-time

Oblique decision trees
- Test condition may involve multiple attributes
- More expressive representation
- Finding optimal test condition is computationally expensive

Tree replication
- Same subtree appears in multiple branches
Rule-based classification

Elena Baralis
Politecnico di Torino

Rule-based classifier

- Classify records by using a collection of “if...then...” rules
- Rule: \((\text{Condition}) \rightarrow y\)
  - \(y\) is the class label
  - \(\text{LHS}: \text{rule antecedent or condition}\)
  - \(\text{RHS}: \text{rule consequent}\)
- Examples of classification rules
  - \((\text{Blood Type}=\text{Warm}) \land (\text{Live in Water}=\text{yes}) \rightarrow \text{Birds}\)
  - \((\text{Taxable Income}< 50\text{K}) \land (\text{Refund}=\text{Yes}) \rightarrow \text{Evade=No}\)

Rule-based Classifier (Example)

<table>
<thead>
<tr>
<th>Name</th>
<th>Blood Type</th>
<th>Give Birth</th>
<th>Can Fly</th>
<th>Live in Water</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>eagle</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>grizzly bear</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>human</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>owl</td>
<td>cold</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>whale</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>python</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>turtle</td>
<td>sometimes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>whale</td>
<td>warm</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>

Rule R1 covers a hawk => Bird
Rule R3 covers the grizzly bear => Mammal

Rule-based classification

- A rule \(\text{covers}\) an instance \(x\) if the attributes of the instance satisfy the condition of the rule
- \(\text{R}_1: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{yes}) \rightarrow \text{Birds}\)
- \(\text{R}_2: (\text{Give Birth} = \text{no}) \land (\text{Live in Water} = \text{yes}) \rightarrow \text{Fishes}\)
- \(\text{R}_3: (\text{Give Birth} = \text{yes}) \land (\text{Blood Type} = \text{warm}) \rightarrow \text{Mammals}\)
- \(\text{R}_4: (\text{Give Birth} = \text{no}) \land (\text{Can Fly} = \text{no}) \rightarrow \text{Reptiles}\)
- \(\text{R}_5: (\text{Live in Water} = \text{sometimes}) \rightarrow \text{Amphibians}\)

Characteristics of rules

- Mutually exclusive rules
  - Classifier contains mutually exclusive rules if the rules are independent of each other
  - Every record is covered by at most one rule
- Exhaustive rules
  - Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
  - Each record is covered by at least one rule

A lemur triggers R3, so it is classified as a mammal
A turtle triggers both R4 and R5
A dogfish shark triggers none of the rules
Data mining: classification

From decision trees to rules

Classical rules

- Initial Rule: (Refund=No) \rightarrow \text{No}
- Simplified Rule: (Status=Married) \rightarrow \text{No}

Rules are mutually exclusive and exhaustive
- Rule set contains as much information as the tree

Effect of rule simplification

- Rules are no longer mutually exclusive
  - A record may trigger more than one rule
  - Solution?
    - Ordered rule set
      - Use voting schemes
  - Rules are no longer exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

Ordered rule set

- Rules are rank ordered according to their priority
- An ordered rule set is known as a decision list
- When a test record is presented to the classifier
  - It is assigned to the class label of the highest ranked rule it has triggered
  - If none of the rules fired, it is assigned to the default class

Building classification rules

- Direct Method
  - Extract rules directly from data
  - e.g.: RIPPER, CN2, Holte's 1R
- Indirect Method
  - Extract rules from other classification models (e.g. decision trees, neural networks, etc.)
  - e.g.: C4.5rules

Advantages of rule-based classifiers

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees
Associative classification

The classification model is defined by means of association rules

\[( \text{Condition} ) \rightarrow y \]

- rule body is an itemset
- Model generation
  - Rule selection & sorting
    - based on support, confidence and correlation thresholds
  - Rule pruning
    - Database coverage: the training set is covered by selecting topmost rules according to previous sort

Strong points
- interpretable model
- higher accuracy than decision trees
- correlation among attributes is considered
- efficient classification
- unaffected by missing data
- good scalability in the training set size

Weak points
- rule generation may be slow
  - it depends on support threshold
  - reduced scalability in the number of attributes
  - rule generation may become unfeasible

Neural networks

Inspired to the structure of the human brain
- Neurons as elaboration units
- Synapses as connection network

Structure of a neural network

Data mining: classification

Structure of a neuron

![Structure of a neuron](image)

Construction of the neural network

- For each node, definition of
  - set of weights
  - offset value
  - providing the highest accuracy on the training data
- Iterative approach on training data instances

Construction of the neural network

- Base algorithm
  - Initially assign random values to weights and offsets
  - Process instances in the training set one at a time
    - For each neuron, compute the result when applying weights, offset and activation function for the instance
    - Forward propagation until the output is computed
    - Compare the computed output with the expected output, and evaluate error
    - Backpropagation of the error, by updating weights and offset for each neuron
  - The process ends when
    - % of accuracy above a given threshold
    - % of parameter variation (error) below a given threshold
    - The maximum number of epochs is reached

Neural networks

- Strong points
  - High accuracy
  - Robust to noise and outliers
  - Supports both discrete and continuous output
  - Efficient during classification
- Weak points
  - Long training time
    - weakly scalable in training data size
    - complex configuration
  - Not interpretable model
    - application domain knowledge cannot be exploited in the model

Bayesian Classification

Bayes theorem

- Let C and X be random variables
  \[ P(C,X) = P(C|X) P(X) \]
  \[ P(C,X) = P(X|C) P(C) \]
- Hence
  \[ P(C|X) P(X) = P(X|C) P(C) \]
- and also
  \[ P(C|X) = P(X|C) P(C) / P(X) \]
Bayesian classification

- Let the class attribute and all data attributes be random variables
  - C = any class label
  - X = \(<x_1,...,x_k>\) record to be classified
- Bayesian classification
  - compute P(C|X) for all classes
  - probability that record X belongs to C
  - assign X to the class with maximal P(C|X)
- Applying Bayes theorem
  \[ P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \]
  - P(X) constant for all C, disregarded for maximum computation
  - P(C) a priori probability of C
    \[ P(C) = \frac{N_c}{N} \]
  - P(x_1,...,x_k|C) = \frac{|x_kC|}{N_c} \]
    - for discrete attributes
    - for continuous attributes, use probability distribution

Bayesian classification: Example

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>N</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>N</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>N</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>P</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>N</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>P</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>P</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>P</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>N</td>
</tr>
</tbody>
</table>

Support Vector Machines

Elena Baralis
Politecnico di Torino
Support Vector Machines

- Find a linear hyperplane (decision boundary) that will separate the data.

One Possible Solution

Another possible solution

Other possible solutions

Which one is better? B1 or B2?
- How do you define better?

Find hyperplane maximizes the margin => B1 is better than B2

From: Tan, Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006
Nonlinear Support Vector Machines

- What if decision boundary is not linear?

Nonlinear Support Vector Machines

- Transform data into higher dimensional space

Instance-Based Classifiers

Set of Stored Cases

<table>
<thead>
<tr>
<th>Attr1</th>
<th>AttrN</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
</tr>
</tbody>
</table>

Unseen Case

- Store the training records
- Use training records to predict the class label of unseen cases

K-Nearest Neighbor

Elena Baralis
Politecnico di Torino

Instance Based Classifiers

- Examples
  - Rote-learner:
    - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
  - Nearest neighbor:
    - Uses k “closest” points (nearest neighbors) for performing classification

Nearest-Neighbor Classifiers

- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of k, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify k nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)
**Definition of Nearest Neighbor**

K-nearest neighbors of a record \( x \) are data points that have the k smallest distance to \( x \)

- (a) 1-nearest neighbor
- (b) 2-nearest neighbor
- (c) 3-nearest neighbor


---

**Nearest Neighbor Classification**

- Compute distance between two points:
  - Euclidean distance
    \[
    d(p, q) = \sqrt{\sum (p_i - q_i)^2}
    \]
- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, \( w = 1/d^2 \)


---

**Choosing the value of k:**

- If k is too small, sensitive to noise points
- If k is too large, neighborhood may include points from other classes

---

**Nearest Neighbor Classification**

- Scaling issues
  - Attribute domain should be normalized to prevent distance measures from being dominated by one of the attributes
  - Example: height [1.5m to 2.0m] vs. income [$10K to $1M]
  - Problem with distance measures
    - High dimensional data
      - curse of dimensionality

---

**Model evaluation**

Elena Baralis
Politecnico di Torino
Data mining: classification

Model evaluation
- Methods for performance evaluation
  - Partitioning techniques for training and test sets
- Metrics for performance evaluation
  - Accuracy, other measures
- Techniques for model comparison
  - ROC curve

Methods for performance evaluation
- Objective
  - reliable estimate of performance
- Performance of a model may depend on other factors besides the learning algorithm
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning curve
- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)
- Effect of small sample size:
  - Bias in the estimate
  - Variance of estimate

Methods of estimation
- Partitioning labeled data in
  - training set for model building
  - test set for model evaluation
- Several partitioning techniques
  - holdout
  - cross validation
- Stratified sampling to generate partitions
  - without replacement
  - Bootstrap
  - Sampling with replacement

Holdout
- Fixed partitioning
  - reserve 2/3 for training and 1/3 for testing
- Appropriate for large datasets
  - may be repeated several times
    - repeated holdout

Cross validation
- Cross validation
  - partition data into k disjoint subsets (i.e., folds)
  - k-fold: train on k-1 partitions, test on the remaining one
  - repeat for all folds
  - reliable accuracy estimation, not appropriate for very large datasets
- Leave-one-out
  - cross validation for k=n
  - only appropriate for very small datasets
Data mining: classification

Metrics for model evaluation

- Evaluate the predictive accuracy of a model
- Confusion matrix
- Binary classifier

<table>
<thead>
<tr>
<th>ACTUAL CLASS</th>
<th>PREDICTED CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class=Yes</td>
<td>a</td>
</tr>
<tr>
<td>Class=No</td>
<td>c</td>
</tr>
</tbody>
</table>

Accuracy

- Most widely used metric for model evaluation

\[
\text{Accuracy} = \frac{\text{Number of correctly classified objects}}{\text{Number of classified objects}}
\]

- Not always a reliable metric

Accuracy for a binary classifier

\[
\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Limitations of accuracy

- Classes may have different importance
  - Misclassification of objects of a given class is more important
  - e.g., ill patients erroneously assigned to the healthy patients class
  - Accuracy is not appropriate for
    - unbalanced class label distribution
    - different class relevance

Class specific measures

- Evaluate separately for each class

\[
\text{Recall} (r) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects belonging to } C}
\]

\[
\text{Precision} (p) = \frac{\text{Number of objects correctly assigned to } C}{\text{Number of objects assigned to } C}
\]

\[
\text{Maximize}
\]

\[
F\text{-measure} (F) = \frac{2rp}{r + p}
\]

Elena Baralis
Politecnico di Torino
Data mining: classification

Class specific measures

- For a binary classification problem on the confusion matrix, for the positive class
  \[
  \text{Precision}(p) = \frac{a}{a + c}
  \]
  \[
  \text{Recall}(r) = \frac{a}{a + b}
  \]
  \[
  F\text{-}measure(F) = \frac{2pr}{r + p} = \frac{2a}{2a + b + c}
  \]

Commentary

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
- characterizes the trade-off between positive hits and false alarms
- ROC curve plots
  - TPR, True Positive Rate (on the y-axis)
    \[
    \text{TPR} = \frac{TP}{TP + FN}
    \]
  - FPR, False Positive Rate (on the x-axis)
    \[
    \text{FPR} = \frac{FP}{FP + TN}
    \]

Commentary

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

ROC curve

(FPR, TPR)

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal

Commentary

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

How to build a ROC curve

- Use classifier that produces posterior probability for each test instance \(P(+|A)\)
- Sort the instances according to \(P(+|A)\) in decreasing order
- Apply threshold at each unique value of \(P(+|A)\)
- Count the number of TP, FP, TN, FN at each threshold
- TP rate
  \[
  \text{TP rate} = \frac{TP}{TP + FN}
  \]
- FP rate
  \[
  \text{FP rate} = \frac{FP}{FP + TN}
  \]

Commentary

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006

Using ROC for Model Comparison

- No model consistently outperforms the other
- \(M_1\) is better for small FPR
- \(M_2\) is better for large FPR
- Area under ROC curve
  - Ideal
    \[
    \text{Area} = 1.0
    \]
  - Random guess
    \[
    \text{Area} = 0.5
    \]

Commentary

From: Tan,Steinbach, Kumar, Introduction to Data Mining, McGraw Hill 2006