Classification and Prediction

Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary
Tasks

- **Classification & Prediction**
  - describe data with respect to a target class/output
  - create, from a data set, a model that
    - describes a target class in terms of features (or attributes or variables) of the class or system generating the data
  - use the model to
    - classify an unknown target class
    - predict an unknown output

- **Example Applications**
  - credit approval
  - medical diagnosis
  - treatment effectiveness analysis

*What about image recognition or customer profiling?*

---

**Classification vs. Prediction**

![Diagram of classification vs. prediction](image-url)

- **Comparisons:**
  - **Data**
    - Labeled
    - Data with unknown class label or output
  - **Model**
    - Algorithm
    - Model
    - class label or predicted output
  - **Use of the model**
    - Labeled (by known target class)
      - classifier/classification model classify unknown label categorical
    - Labeled (by known output)
      - predictive model predict unknown output continuous/ordered
Three-step process

Model Construction (Training/Learning)
- Build a model that describes training data labeled with known classes/outputs
- A model could be of various forms: rules, decision trees, formulae

Model Evaluation (Testing)
- Use a labeled testing data set to estimate model accuracy
  - The known label of test sample is compared with the classified result from the model
  - Accuracy = % test cases correctly classified by the model

Model Usage (Classification)
- Use the model with acceptable accuracy to classify or predict unknown

Example

<table>
<thead>
<tr>
<th>Inst</th>
<th>Credit history</th>
<th>Debt</th>
<th>Collateral</th>
<th>Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>U</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>Bad</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>Bad</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>Good</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>10</td>
<td>Good</td>
<td>H</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>11</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>12</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>14</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>

Classification Algorithm

Model Construction

Classifier (Model):
1. If income = L then Risk = H
2. If income = H and Credit History = U then Risk = L
   ........

Model Evaluation

Accuracy = ?

Model Usage

<table>
<thead>
<tr>
<th>Inst</th>
<th>Credit history</th>
<th>Debt</th>
<th>Collateral</th>
<th>Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U</td>
<td>H</td>
<td>Ok</td>
<td>L</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>Bad</td>
<td>L</td>
<td>None</td>
<td>M</td>
<td>?</td>
</tr>
</tbody>
</table>
Machine learning context

Training Data → Algorithm → Model → Is the model sufficiently accurate?
- no
- yes

Data with unknown class label or output → Output Model
class label or predicted output

Supervised vs. Unsupervised

- **Supervised learning**
  - Class labels and # classes are known
  - Labeled data: the training data (observations, measurements, features etc.) are accompanied by labels indicating the class of the observations
  - Goal is to describe class/concept in terms of features/observations
    → classification/prediction

- **Unsupervised learning**
  - Class labels are unknown and # classes may not be known
  - Unlabeled data: the training data does not have class labels
  - Goal is to establish the existence of classes or concepts or clusters in the data
    → clustering
Terms

- Machine learning aims to abstract regularities from a given data set
  - Mining ~ learning ~ induction (reasoning from specifics to general)
- A training set consists of training instances
  - The terms instances, tuples (or rows), examples, samples, and objects are used synonymously
- A testing set is independent of a training set
  - If a training set is used for testing during learning process, the resulting model will overfit the training data
    - the model could incorporate anomalies in the training data
      - that are not present in the overall sample population
    - model that may not represent the population (poor generalization)

Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary
Evaluation

- A resulting model (classifier) is tested against a testing data set to measure its
  - Accuracy or
  - Error rate = # misclassified test cases/#test cases in %
- Experimental methods
  - Holdout method
  - Random subsampling
  - Cross validation
  - Bootstrapping

Holdout method

- Randomly partition a data set into two independent sets: a training set and a test set (typically 2/3 and 1/3)
  - The training set is used to derive a classifier, whose accuracy is estimated using the test set
- Use with data set that has large number of samples
Random sub-sampling

- Repeat holdout method \( k \) times
- Overall accuracy = average of the accuracies in each iteration

Cross validation

- \( k \)-fold cross validation
  - Divide a data set into \( k \) approximately equal subsets of samples
  - Use \( k-1 \) subsets for training set and the rest for testing set
  - Obtain a classifier/accuracy from each pair of training and testing sets (\( k \) possibilities)
Cross validation & Bootstrapping

- **Stratified cross validation** – the folds are stratified so that the class distribution of the samples in each fold is approximately the same as that in the initial data.

- Stratified 10-fold Cross-validation is recommended when a sample set is of medium size.

- When sample set size is very small → **Bootstrapping**
  - Leave-one-out sample training set with replacement

Improving Accuracy

- Use a committee of classifiers and combine the results of each classifier.

- Two basic techniques:
  - Bagging
  - Boosting
Bagging

- Generate **independent training sets** by random sampling with replacement from the original sample set.
- Construct a **classifier for each training set** using the same classification algorithm.
- To classify an unknown sample $x$, each classifier returns its class prediction, which counts as one vote.
- The Bagged Classifier $C^*$ counts the votes and assigns $x$ to the class with the “most” votes.
Boosting

- Assign weight to each training sample
- A series of classifiers is learned. After classifier $C_i$ is learned
  - Calculate the error and re-weight the examples based on the error.
  - The weights are updated to allow the subsequent classifier $C_{i+1}$ to “pay more attention” to the misclassification errors made by $C_i$

- The final boosted classifier $C^*$ combines the votes of each classifier, where the weight of each classifier’s vote is a function of its accuracy on the training set
Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary

Issues on classification/prediction

- How do we compare different techniques?
  - No single method works well for all data sets

- Some evaluation criteria:
  - **Accuracy** – indicates generalization power
  - **Speed** – time to train/learn a model and time to use the model
  - **Robustness** – tolerant to noise and incorrect data
  - **Scalability** – acceptable run time to learn as data size and complexity grow
  - **Interpretability** – comprehensibility and insight from the resulting model
Issues on classification/prediction

- Any alternatives to the accuracy measure?
  - Sensitivity = $t_{pos}/pos$
  - Specificity = $t_{neg}/neg$
  - Accuracy = sensitivity $(pos/(pos+neg)) + specificity (neg/(pos+neg))$
    \[= (t_{pos} + t_{neg})/(pos+neg)\]

<table>
<thead>
<tr>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancer</td>
<td>$t_{pos}$</td>
</tr>
<tr>
<td>~ cancer</td>
<td>$f_{neg}$</td>
</tr>
</tbody>
</table>

pos = # positive samples
neg = # negative samples
t-pos = # of true positives
t-neg = # of true negatives
f-pos = # of false positives
f-neg = # of false negatives

Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary
Decision Tree (DT) Classification

- Typically has two phases:
  - Tree construction
  - Tree pruning

- Tree construction (Induction)
  - Initially, all the training examples are at the root
  - Recursively partition examples based on selected attributes

- Tree pruning
  - Remove branches that may reflect noise/outliers in the data
  - Give faster or more accurate classification

Input/Output

Training data set

Does the tree and the data set cover the same information?
Decision Tree Induction

Basic Algorithm
- Data are categorical
- Tree starts a single node representing all data
- Recursively,
  - select a split-attribute node that best separates sample classes
  - If all samples for a given node belong to the same class
    → the node becomes a leaf labeled with class label
  - If no remaining attributes on which samples may be partitioned
    Or there are no samples for the attribute value branch
    → a leaf is labeled with majority class label in samples

Example

<table>
<thead>
<tr>
<th>inst</th>
<th>Credit History</th>
<th>Debt</th>
<th>Collateral</th>
<th>Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>U</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>Bad</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>Bad</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>Good</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>10</td>
<td>Good</td>
<td>H</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>11</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>12</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>14</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>inst</th>
<th>Credit history</th>
<th>Debt</th>
<th>Collateral</th>
<th>Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>U</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>Bad</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>Bad</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>Good</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>10</td>
<td>Good</td>
<td>H</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>11</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>12</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>14</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Inst</th>
<th>Credit history</th>
<th>Debt</th>
<th>Collateral</th>
<th>Income</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>4</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>5</td>
<td>U</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td>U</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>7</td>
<td>Bad</td>
<td>L</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>8</td>
<td>Bad</td>
<td>L</td>
<td>Ok</td>
<td>H</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>Good</td>
<td>L</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>10</td>
<td>Good</td>
<td>H</td>
<td>Ok</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>11</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>12</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>M</td>
</tr>
<tr>
<td>13</td>
<td>Good</td>
<td>H</td>
<td>None</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>14</td>
<td>Bad</td>
<td>H</td>
<td>None</td>
<td>M</td>
<td>H</td>
</tr>
</tbody>
</table>

#### Selecting split-attribute

- Based on a measure called *Goodness function*
- Different algorithms use different goodness functions:
  - **Information gain** (e.g. in ID3/C4.5 [Quinlan, 94])
    - Assume categorical attribute values - modifiable to continuous-valued attributes
    - Choose attribute with highest information gain
  - **Gini Index** (e.g., in CART, IBM IntelligentMiner)
    - Assume continuous-valued attributes – modifiable to categorical attributes
    - Assume several possible split values for each attribute – may need clustering to find possible split values
    - Choose attribute with *smallest* gini index
  - **Others** e.g., gain ratio, distance-based measure, etc.
Information gain

- Choose attribute with highest information gain to be the test attribute for current node
- Highest information gain = greatest entropy reduction
  - least randomness in sample partitions
  - min information needed to classify the sample partitions

Information gain

Let $S$ be a set of labeled samples, then the information needed to classify a given sample is

$$Ent(S) = - \sum_{i \in C} p_i \log_2 p_i$$

where $p_i$ is a probability that a sample from $S$ is in class $i$. Here $p_i = |C_i| / |S|$, where $C_i$ is a sample set from $S$ whose class label is $i$.

Expected information needed to classify a sample if it is partitioned into subsets by $A$.

$$I(A) = \sum_{i \in \text{dom}(A)} \frac{|S_i|}{|S|} Ent(S_i)$$

where $S_i$ is a partition of a sample set whose $A$ attribute value is $i$, and $\text{dom}(A) = \text{all possible values of } A$.

Information gain:

$$\text{Gain}(A) = Ent(S) - I(A)$$
Example

Goal - to find attribute $A$ with highest gain$(A)$ on $S$

$S = \{1, 2, 3, ..., 14\}$ - each number represent a sample row

$C = \{H \text{ Risk}, M \text{ Risk}, L \text{ Risk}\}$

$(1, 2, 4, 7, 11, 14)$ $(3, 8, 12)$ $(5, 6, 9, 10, 13)$

$\text{Gain}(\text{income}) = \text{Ent}(S) - \text{Ent}(\text{income})$

Similarly,

$\text{Gain(credit history)} = 0.266$

$\text{Gain(debt)} = 0.581$

$\text{Gain(collateral)} = 0.756$

$\rightarrow$ Select income as a root
Gini Index

- For a data set $T$ containing examples from $n$ classes, define
  \[ gini(T) = 1 - \sum_{j=1}^{n} p_j^2 \]
  where $p_j$ is the relative frequency of class $j$ in $T$

- If a data set $T$ is split into two subsets $T_1$ and $T_2$, the gini index of the split data contains examples from $n$ classes is defined as
  \[ gini_{split}(T) = \frac{|T_1|}{N} gini(T_1) + \frac{|T_2|}{N} gini(T_2) \]

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (need to enumerate all possible splitting points for each attribute).

DT induction algorithm

Characteristics:

- Greedy search algorithm
  - Make optimal choice at each step – select the “best” split-attribute for each tree node

- Top-down recursive divide-and-conquer manner
  - From root to leaf
  - Split node to several branches and each branch, recursively run algorithm to build a subtree
DT Construction Algorithm

Primary Issues:
- **Split criterion** – for selecting split-attribute
  - Different algorithms use different goodness functions: information gain, gini index etc.
- **Branching scheme** – how many sample partitions
  - Binary branches (gini index) vs. multiple branches (information gain)
- **Stopping condition** – when to stop further splitting
  - Fully-grown vs. stopping early
- **Labeling class**
  - Node is labeled with the most common class

From Trees to Rules

- Represent the knowledge in the form of **IF-THEN** rules
- Create one rule for each path from the root to a leaf
  - Each attribute-value pair along a path forms a conjunction
  - The leaf node holds the class prediction
- Rules are easier for humans to understand

Example:

**IF** income = "M" AND credit-history = "U" AND debt = "H" **THEN** risk = "H"

**IF** income = "M" AND credit-history = "Good" **THEN** risk = "M"
Overfitting & Tree Pruning

- A tree constructed may overfit the training examples due to noise or too small a set of training data → Poor accuracy for unseen samples
- Approaches to avoid overfitting
  - **Prepruning**: Stop growing the tree early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning**: Remove branches from a “fully grown” tree—get a sequence of progressively pruned trees then decide which is the “best pruned tree”
    - Combined
- Postpruning is more expensive than prepruning but more effective

---

Postpruning

- Merge a subtree into a leaf node
  - If accuracy without splitting > accuracy with splitting, then don’t split
    ⇒ replace the subtree with a leaf node, label it with a majority class
- Techniques:
  - **Cost complexity pruning** – based on expected error rates
    - For each non-leaf node in the tree, compute expected error rate if the subtree was and was not pruned – combined error rates from each branch using weights on sample frequency ratios
    - Requires an independent test sample to estimate the accuracy of each of the progressively pruned trees
  - **Min description length pruning** – based on # of tree encoding bits
    - The “best pruned tree” minimizes the # of encoding bits
    - No test sample set is required
Postpruning (cont)

The correct tree size can be determined by:

- Using a **separate test set** to evaluate the pruning
  E.g., CART

- Using all the data for training but
  - applying a **statistical test** (e.g., chi-square) to evaluate the pruning
    E.g., C4.5

- Using minimum description length (**MDL**) principle
  - halting growth of the tree when the encoding is minimized
    E.g., SLIQ, SPRINT

Tree induction Enhancement

- **Allow for continuous-valued attributes**
  - Dynamically define a discrete value that partitions the continuous values into a discrete set of intervals
    - sort continuous attribute values, identify adjacent values with different target classes, generate candidate thresholds midway, and select the one with max gain

- **Handle missing attribute values**
  - Assign the most common value OR probability to each possible values

- **Reduce tree fragmentation, testing repetition, and subtree replication**
  - **Attribute construction** - create new attributes based on existing ones that are sparsely represented
Classification in Large Databases

- Classification: a classical problem extensively studied by statisticians and machine learning researchers
- Goal: classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Decision tree induction seems to be a good candidate
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can be used to generate SQL queries for accessing databases
  - has comparable classification accuracy with other methods
- Primary issue: scalability
  - most algorithms assume data can fit in memory – need to swap data in/out of main and cache memories

Scaling up decision tree methods

- **Incremental tree construction** [Quinlan, 86]
  - Using partial data to build a tree
  - Testing of additional examples and examples that are misclassified are used to rebuild the tree interactively
- **Data reduction** [Cattlet, 91] – still a main memory algorithm
  - Reducing data size by sampling and discretization
- **Data partition and merge** [Chan and Stolfo, 91]
  - Building trees for each partition of the data
  - Merging trees into one tree
  - But … resulting accuracy is reported to be reduced
Recent efforts in Data Mining Studies

- **SLIQ** (EDBT ’96 — Mehta et al.) & **SPRINT** (VLDB ’96 — J. Shafer et al.)
  - presort disk-resident data sets (that are too large to fit in main memory)
  - handle continuous-valued attributes
  - define a new data structure to facilitate tree construction

- **PUBLIC** (VLDB ’98 — Rastogi & Shim)
  - integrates tree splitting and tree pruning

- **RainForest** (VLDB ’98 — Gehrke, Ramakrishnan & Ganti)
  - A framework for scaling decision tree induction that separates scalability from quality criteria

---

**SLIQ**

- Tree building uses a data structure: **attribute lists, a class list**
- Each attribute has an associated attribute list, indexed by record ID
- Each tuple is a linkage from each attribute list to a class list entry and to a leaf node of the decision tree

<table>
<thead>
<tr>
<th>RID</th>
<th>credit-rate</th>
<th>age</th>
<th>buy-car?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>good</td>
<td>30</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>fair</td>
<td>28</td>
<td>no</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>credit-rate</th>
<th>RID</th>
<th>age</th>
<th>RID</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>1</td>
<td>26</td>
<td>2</td>
</tr>
<tr>
<td>fair</td>
<td>2</td>
<td>30</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RID</th>
<th>buy-car?</th>
<th>node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
SLIQ

Efficiency is contributed by the fact that:
- Only class list and current attribute list are memory-resident
  - Class list is dynamically modified during tree construction
- Attribute lists are disk-resident, presorted (reduce cost to evaluate splitting) and indexed (eliminate resorting data)
- Use fast subsetting algorithm to determine split-attributes
  - If number of possible subsets exceeds threshold, use greedy search
- Use inexpensive MDL-based tree pruning

SPRINT

- Proposes a new data structure: attribute list eliminating SLIQ’s requirement on memory-resident class list
  - Attribute 1        class                                   Attribute 2     class
    RID   credit-rate  age  buy-car?
    1     good         30   yes
    2     far          28   no
  - Outperforms SLIQ when the class list is too large to fit main memory
    - but needs a hash tree to connect different joins which could be costly with a large training set
    - Designed to ease parallelization
PUBLIC

- Integrates splitting and pruning
- Observation & idea:
  - a large portion of the tree ends up being pruned
  - Can we use a top-down approach to predetermine this and stop growing it earlier?
- How?
  - Before expanding a node, compute a lower bound estimation on the cost subtree rooted at the node
  - If a node is predicted to be pruned (based on the cost estimation), return it as a leaf, otherwise go on splitting it

RainForest

- A generic framework that
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - Applies to any decision tree induction algorithm
  - Maintains an AVC-set (attribute, value, class label)
- Reports a speedup over SPRINT
Data Cube-Based DT Induction

- Integration of generalization with DT induction – once the tree is derived, use a concept hierarchy to generalize
  - Generalization at low-level concepts → large, bushy trees
  - Generalization at high-level concepts → lost interestingness
- Cube-based multi-level classification
  - Relevance analysis at multi-levels

Visualization in Classification

DBMiner - Presentation of classification rules
Visualization in Classification

Interactive Visual Mining by Perception-Based Classification (PBC)
Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary

Bayesian classification

Basic features/advantages:
- Probabilistic classification
  - Gives explicit probability that a given example belongs to a certain class
  - Predict multiple hypotheses, weighted by their probabilities
- Probabilistic learning
  - Each training example can incrementally increase or decrease the probability of the hypothesis
- Probabilities are theoretical-supported
  - Provide standard measures to facilitate decision-making and comparison with other methods
Bayesian-based approaches

- Basic approaches
  - Naïve Bayes Classification
  - Bayesian network learning
- Both are based on Bayes Theorem

Bayes Theorem

- $X$ - an observed training example
- $H$ - a hypothesis that $X$ belongs to a class $C$
- Goal of classification: determine $P(H|X)$, the probability that $H$ holds given $X$.
- **Bayes Theorem:** $P(H|X) = \frac{P(X|H)P(H)}{P(X)}$

  - Informally, this can be written as $\text{posterior} = \text{likelihood} \times \text{prior} / \text{evidence}$ (RHS can be estimated)

  - Practical issues:
    - require initial knowledge of many probabilities
    - significant computational cost
Naïve Bayes Classifier

- a data example $X$, is an $n$-dimensional vector $\rightarrow n$ attributes
- Naïve assumption of class conditional independence
  - For a given class label, attribute values are conditionally independent of one another
  - No dependence relationships between attributes

\[
P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)
\]
where $C_i$ is a class label

- The assumption reduces the computation cost, only count the class distribution.
- Once the probability $P(X \mid C_i)$ is known, assign $X$ to the class with maximum $P(X \mid C_i)P(C_i)$ – Why? and How?

Naïve Bayes Classifier

Once the probability $P(X \mid C_i)$ is known, assign $X$ to the class with maximum $P(X \mid C_i)P(C_i)$ – Why?

The classifier predicts class of $X$ to be the one that gives maximum $P(C_i \mid X)$ for all possible classes $C_i$’s

By Bayes theorem,

\[
P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}
\]

Since $P(X)$ is the same for all classes, only $P(X \mid C_i)P(C_i)$ needs to be maximized
Example

\( X = (\text{age} = <30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit\_rating} = \text{fair}) \)

Compute \( P(X|C_i) \) for each class

\[
P(X|C_i) = \frac{P(X|\text{buys\_car} = \text{yes}) \cdot P(\text{buys\_car} = \text{yes})}{P(X|\text{buys\_car} = \text{yes}) \cdot P(\text{buys\_car} = \text{yes}) + P(X|\text{buys\_car} = \text{no}) \cdot P(\text{buys\_car} = \text{no})}
\]

<table>
<thead>
<tr>
<th>age</th>
<th>income</th>
<th>student</th>
<th>credit_rate</th>
<th>buys_car</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>high</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>fair</td>
<td>no</td>
</tr>
<tr>
<td>&lt;=30</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>low</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>yes</td>
<td>fair</td>
<td>yes</td>
</tr>
<tr>
<td>&gt;40</td>
<td>medium</td>
<td>no</td>
<td>excellent</td>
<td>no</td>
</tr>
</tbody>
</table>

\[
P(X|\text{buys\_car} = \text{yes}) \cdot P(\text{buys\_car} = \text{yes}) = 0.0444 \cdot 9/14 = 0.028
\]

\[
P(X|\text{buys\_car} = \text{no}) \cdot P(\text{buys\_car} = \text{no}) = 0.019 \cdot 5/14 = 0.007
\]

\( X \) belongs to class “\text{buys\_car} = \text{yes}”

Naïve Bayesian Classifiers

- \( P(X|C_j) \) for a continuous-valued attribute, \( A_k \) can be estimated from a normal density function
- Advantages:
  - Easy to implement & good results obtained in most of the cases
- Disadvantages:
  - Loss of accuracy when the class conditional independence assumption is violated
  - In practice, dependencies exist among variables, e.g.,
    - hospitals: patients: Profile: age, family history etc.; symptoms etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks
Bayesian Networks

- Bayesian belief network (Bayes net, Bayesian network) allows class conditional dependencies to be represented.
- A graphical model of dependency among the variables:
  - Gives a specification of joint probability distribution
  - Two components: DAG (Directed Acyclic Graph)
  - CPT (Conditional Probability Table) representing \( P(X \mid \text{parents}(X)) \) for each node.

Nodes: random variables
Links: dependency
\( X, Y \) are the parents of \( Z \)
\( Y \) is the parent of \( P \)
No dependency between \( Z \) and \( P \)
Has no loops or cycles

An Example

- Family History
- Smoker
- LungCancer
- Emphysema
- PositiveXRay
- Dyspnea

CPT of Lung Cancer (LC)

<table>
<thead>
<tr>
<th>(FH, S)</th>
<th>(FH, ~S)</th>
<th>(~FH, S)</th>
<th>(~FH, ~S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>~LC</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
</tr>
</tbody>
</table>

\( P(\text{LC=yes}|\text{FH=yes and S = no}) = 0.5 \)

The joint probability of a tuple \( (z_1, \ldots, z_n) \) corresponding to variables \( Z_1, \ldots, Z_n \) is

\[
P(z_1, \ldots, z_n) = \prod_{i=1}^{n} P(z_i \mid \text{Parents}(Z_i))
\]
Classification with Bayes Nets

- The classification process can return a probability distribution for the class attribute instead of a single class label
- Learning Bayes Nets
  - Causal learning/discovery is not the same as Causal Reasoning
  - Output: A Bayes Net representing regularities in an input data set
  - Network structures may be known or unknown:
    - structure known + all variables observable: training to learn CPTs
    - structure known + some hidden variables:
      - use gradient descent - analogous to neural network learning
    - structure unknown + all variables observable:
      - Search through the model space to reconstruct structure
    - structure unknown + all hidden variables: No good algorithms known
  - Two basic approaches: Bayesian learning and constraint-based learning

Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary
Classification & function approximation

- Classification: predicts categorical class labels
- Typical Applications:
  - \{credit history, salary\} → Credit approval (Yes/No)
  - \{Temp, Humidity\} → Rain (Yes/No)

Mathematically,
\[
x \in X = \{0,1\}^n, y \in Y = \{0,1\}
\]
\[
h : X \rightarrow Y
\]
\[
y = h(x)
\]

We want to estimate function \( h \)

Linear Classification

- Binary Classification problem
  - The data above the line belongs to class ‘x’
  - The data below line belongs to class ‘o’

- Discriminative Classifiers, e.g.,
  - SVM (support vector machine)
  - Perceptron (Neural net approach)
Discriminative Classifiers

- Advantages
  - prediction accuracy is generally high
    as compared to Bayesian methods – in general
  - robust, works when training examples contain errors
  - fast evaluation of the learned target function
    Bayesian networks are normally slow

- Criticism
  - long training time
  - difficult to understand the learned function (weights)
    Bayesian networks can be used easily for pattern discovery
  - not easy to incorporate domain knowledge
    easy in the form of priors on the data or distributions

Neural Networks

- Analogy to Biological Systems
- Massive Parallelism allowing for computational efficiency
- The first learning algorithm came in 1959 (Rosenblatt)
  who suggested that if a target output value is provided
  for a single neuron with fixed inputs, one can
  incrementally change weights to learn to produce these
  outputs using the perceptron learning rule
A Neuron

The $n$-dimensional input vector $x$ is mapped into variable $y$ by means of the scalar product and a nonlinear function mapping, e.g.,

$$y = \text{sign}(\sum_{i=0}^{n} w_i x_i + \theta_k)$$

Multilayer Neural Networks

Three-layer neural network
- **Feed-forward** – no feedback
- **Fully connected**

Multilayer feed-forward networks of linear threshold functions with enough hidden units can closely approximate any function.
Defining a network structure

Before training, ...
- Determine # units in input layer, hidden layer(s), and output layer
  - No clear rules on the “best” number of hidden layer units
- Normalize input values to speedup learning
  - Continuous values $\rightarrow$ range between 0 and 1
  - Discrete values $\rightarrow$ Boolean values
- Accuracy depends on both network topology and initial weights

Training Neural Networks

- Goal
  - Find a set of weights that makes almost all the tuples in the training data classified correctly (or acceptable error rates)
- Steps
  - Initialize weights with random values
  - Feed the input tuples into the network one by one
  - For each unit
    - Compute the net input to the unit as a linear combination of all the inputs to the unit
    - Compute the output value using the activation function
    - Compute the error
    - Update the weights and the bias
  - Repeat until terminal condition is satisfied
Backpropagation

- Learning ~ searching using a gradient descent method
- Searches for a set of weights
  - that minimizes the mean squared distance between the network’s class prediction and the actual class label of the samples
- A learning rate helps avoid local minimum problem
  - If the rate is too low → learning at a slow pace
  - If the rate is too high → oscillation between solutions
  - Rules of thumb: set it to 1/t where t = # iterations through the training set so far

Initialization
- Weights & biases: small random numbers (e.g., from -1 to 1)

For each training example \( x = (x_1, x_2, x_3) = (1, 0, 1) \)
- Propagate the input forward

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

Sigmoid activation function

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

\[
I_4 = w_{41}O_1 + w_{42}O_2 + \theta_4
\]

where \( I_4 = x_1 = 1 \)

\[
I_5 = w_{51}O_1 + w_{52}O_2 + \theta_5
\]

\[
I_6 = w_{61}O_1 + w_{62}O_2 + \theta_6
\]

Then, compute \( O_6 \)
Backpropagation

- Backpropagate the error
  \[ \text{Err}_j = O_j \left( 1 - O_j \right) (T_j - O_j) \]
  \[ \text{Err}_a = O_a \left( 1 - O_a \right) (T_a - O_a), \text{ where } T_a \text{ is a given target class label} = 1 \]
  \[ \text{Err}_j = O_j \left( 1 - O_j \right) \sum_k \text{Err}_k w_{jk} \]

- Update weights & biases
  \[ w_{ij} = w_{ij} + (l)\text{Err}_j O_i \]
  \[ \theta_j = \theta_j + (l)\text{Err}_j \]

- Case updating – for each training example
- Epoch updating – for each epoch (an iteration through a training set)
Backpropagation

- Repeat the training process (i.e., propagation forward etc.)
- Until
  - All the changes of $w_{ij}$ in the previous epoch is below some specified threshold, or
  - % samples misclassified in the previous epoch is below a threshold, or
  - A pre-specified number of epochs has expired

Pruning and Rule Extraction

- A major drawback of Neural Net learning is its model is hard to interpret
- Fully connected network is hard to articulate $\to$ network pruning
  - Remove weighted links that do not decrease classification accuracy of the network
- Extracting rules from a trained pruned network
  - Discretize activation values; replace individual activation value by the cluster average maintaining the network accuracy
  - Derive rules relating activation value and output using the discretized activation values to enumerate the output
  - Similarly, find the rule relating the input and activation value
  - Combine the above two to have rules relating the output to input
Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary

Other Classification Methods

- Support vector machine
- Classification-based association
- K-nearest neighbor classifier
- Case-based reasoning
- Soft computing approach
  - Genetic algorithm
  - Rough set approach
  - Fuzzy set approaches
Support vector machine (SVM).

- **Basic idea**
  - find the best boundary between classes
  - build classifier on top of them
- Boundary points are called **support vectors**
- Two basic types:
  - Linear SVM
  - Non-linear SVM

SVM

![Small Margin vs Large Margin](image)
Optimal Hyperplane: separable case

- Class 1 and class 2 are separable.
- Crossed points are support vectors – points that maximize the margin between the two classes.
- Given a training set of N pairs of $x_i$ with label $y_i$.
- Goal is to maximize $C$ subject to a constraint:
  \[
  y_i(x_i^T \beta + \beta_0) > C, \ i = 1, ..., N
  \]

Non-separable case

- When the data set is non-separable
  - assign weight to each support vector changing the constraint to be:
    \[
    y_i(x_i^T \beta + \beta_0) > C(1 - \xi_i),
    \]
    where $\forall i, \xi_i > 0$, $\sum_{i=1}^{N} \xi_i < \text{const.}$
General SVM

- No good optimal linear classifier
- Can we do better?
  - A non-linear boundary
    (circled points are support vectors)
- Idea:
  - Map original problem space into a larger space so that the boundary in the new space is linearly separable
  - Use Kernel (do not always exist) to compute distances in the new space (could be infinite-dimensional)

Example

- When the data is not linearly separable
  - Project the data to high dimensional space where it is linearly separable and then we can use linear SVM
Performance of SVM

- For general support vector machine
  \[ E[P(\text{error})] \leq E[\text{(\# of support vectors)}/(\# \text{ training samples})] \]
- SVM has been very successful in lots of applications

SVM vs. Neural Network

- SVM
  - Relatively new concept
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- Neural Network
  - Quite Old
  - Generalizes well but doesn’t have strong mathematical foundation
  - Can easily be learned in incremental fashion
  - To learn complex functions – use multilayer perceptron (non-trivial)
Open problems of SVM

- How to choose appropriate Kernel function
  - Different Kernel gives different results, although generally better than hyper planes
- For very large training set, support vectors might be of large size. Speed thus becomes a bottleneck
- An optimal design for multi-class SVM classifier

SVM Related Links

- [http://svm.dcs.rhbnc.ac.uk/](http://svm.dcs.rhbnc.ac.uk/)
- C. J. C. Burges.
- SVM_light – Software (in C) [http://ais.gmd.de/~thorsten/svm_light](http://ais.gmd.de/~thorsten/svm_light)
Other Classification Methods

- Support vector machine
- Classification-based association
- K-nearest neighbor classifier
- Case-based reasoning
- Soft computing approach
  - Genetic algorithm
  - Rough set approach
  - Fuzzy set approaches

Association-Based Classification

- Several methods
  - ARCS: Association rule + clustering (Lent et al’ 97)
    - It beats C4.5 in (mainly) scalability and also accuracy
  - Associative classification: (Liu et al’ 98)
    - It mines high support and high confidence rules in the form of “cond_set => y”, where y is a class label
  - CAEP (Classification by aggregating emerging patterns) (Dong et al’ 99)
    - Emerging patterns (EPs): the itemsets whose support increases significantly from one class to another
    - Mine EPs based on minimum support and growth rate
Instance-Based Methods

- Instance-based learning:
  - Delay generalization until new instance must be classified model ("lazy evaluation" or "lazy learning")
  - Note: "eager learning" (e.g., decision tree) constructs generalization model before classifying a new sample
- Thus, lazy learning gives fast training but slow classification
- Examples of lazy learning:
  - k-nearest neighbor approach
  - Locally weighted regression
  - Case-based reasoning
- Requires efficient indexing techniques

k-nearest neighbor classifiers

- A training sample = point in n-dimensional space
- When an unknown sample $x$ is given for classification
  - Search for $k$ training samples ($k$ "nearest neighbors") that are closest to $x$
    - E.g., "closeness" ~ Euclidean distance
  - For discrete value class, $x$ is assigned to the most common class among its $k$ "nearest neighbors"
  - For continuous-valued class (e.g., in prediction), $x$ is assigned to the average value of the continuous-valued class associated with the $k$ nearest neighbors
- Potentially expensive when number of k-nearest neighbors is large
**k-nearest neighbor classifiers (cont)**

- **Distance-weighted nearest neighbor algorithm**
  - Weight the contribution of each of the $k$ neighbors according to their distance to the query point $x_q$ - e.g., $w = \frac{1}{d(x_q,x_i)^2}$ giving greater weight to closer neighbors.
  - Similarly, for real-valued target functions.
- Robust to noisy data by averaging $k$-nearest neighbors.
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.
  - To overcome it, axes stretch or elimination of the least relevant attributes.

**Case-Based Reasoning (CBR)**

- Similar to k-nearest neighbor
  - Instance-based learner (lazy learner) & Analyse similar samples
  - But …A sample is a “case” (complex symbolic description) not a point in Euclidean space.
  - Idea: given a new case to classify
    - CBR searches a training case that is identical → gives the stored solution
    - Otherwise find cases with similar components → modify stored solutions to be a solution for a new case.
- Requires background knowledge, tight coupling between case retrieval, knowledge-based reasoning and problem solving.
- Challenges:
  - Similarity metrics
  - Efficient techniques for indexing training cases and combining solutions.
Genetic Algorithms (GA)

- Based on ideas of natural evolution
  - Create an initial population consisting of randomly generated rules - represented by bit strings
  - Generate rule offsprings by applying genetic operators
    - Crossover - swap substrings from pairs of rules
    - Mutation - randomly selected bits are inverted
  - Form a new population containing the fittest rules
    - The fitness of a rule is measured by its classification accuracy on a set of training examples
  - Repeat process until the population is “evolved” to satisfy fitness threshold
- GA facilitates parallelization and popular for solving optimization problems

Rough Set Approach

- Rough sets are used for “roughly” define equivalent classes in the training data set
- A rough set for a given class C is approximated by
  - a lower approximation (samples certain to be in C)
  - an upper approximation (samples indescribable as not be in C)
- Rough sets can also be used for relevance analysis
  - Finding the minimal subsets of attributes is NP-hard

[Diagram of rectangles representing equivalent classes]
Fuzzy Set

- Fuzzy logic
  - uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Fuzzy logic allows classification of high level abstraction
  - Attribute values are converted to fuzzy values
    - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
  - For a given new sample, more than one fuzzy value may apply
  - Each applicable rule contributes a vote for membership in the categories
  - Typically, the truth values for each predicted category are summed/combined

Outline

- Overview
  - Tasks
  - Evaluation
  - Issues
- Classification techniques
  - Decision Tree
  - Bayesian-based
  - Neural network
  - Others
- Prediction techniques
- Summary
What Is Prediction?

- Prediction is similar to classification
  - First, construct a model then use it to predict unknown value
  - Most deal with modeling of continuous-valued functions
  - Major method for prediction is regression
    - Linear and multiple regression
    - Non-linear regression

Regression Models

- **Linear regression**: Data are modeled using a straight line e.g.,
  - $Y = \alpha + \beta X$, where $Y$ = response variable, $X$ = predictor variable
  - Assume variance of $Y$ to be constant
  - $\alpha$ and $\beta$ are regression coefficients are to be estimated
  - using the least squares method (errors between actual data and estimated line are minimized)

- **Multiple regression**: Extended linear model to more than one predictor value, i.e., $Y = b0 + b1\, X1 + b2\, X2$

- **Nonlinear regression**: many can be transformed into linear

- **Log-linear models**: approximate **discrete** probability distributions
Summary

- Classification is an extensively studied problem (mainly in statistics, machine learning & neural networks)
- Classification is probably one of the most widely used data mining techniques with a lot of extensions
- Scalability is still an important issue for database applications: thus combining classification with database techniques should be a promising topic
- Research directions: classification of non-relational data, e.g., text, spatial, multimedia, etc..

References

References (cont)

- W. Li, J. Han, and J. Pei. CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. Proc. 2001 Int. Conf. on Data Mining (ICDM'01), San Jose, CA, Nov. 2001.

References (cont)