Course Schedule - 2007/08

Data Mining and Knowledge Discovery (DM)

• 21 October 2008 15-19 Lectures (Lavrač)
• 22 October 2008 15-19 Practice (Kralj Novak)
• 11 November 2008 15-19 Lectures (Lavrač)
• 12 November 2008 15-19 Practice (Kralj Novak)
• 1 December 2008 16-17 written exam - theory
• 8 December 2008 15-17 seminar topics presentations
• 14 January 2009 15-19 seminar presentations (exam ?)
• Spare date, if needed:
  (28 January 2009 15-19 seminar presentations ?, exam ?)

http://kt.ijs.si/petra_kralj/IPSKnowledgeDiscovery0809.html

DM - Credits and coursework

"New Media and eScience" / "Statistics"

• 12 credits (30 hours / 36 hours)
• Lectures
• Practice
  – Theory exercises and hands-on (WEKA)
• Seminar – choice:
  – Data analysis of your own data (e.g., using WEKA for questionnaire data analysis)
  – Programming assignment - write your own data mining module, and evaluate it on a (few) domain(s)
• Contacts:
  – Nada Lavrač nada.lavrac@ijs.si
  – Petra Kralj Novak petra.kralj@ijs.si

DM - Course Outline

I. Introduction
   – Data Mining and KDD process
   – DM standards, tools and visualization
   – Classification of Data Mining techniques: Predictive and descriptive DM
     (Mladenić et al. Ch. 1 and 11, Kononenko & Kukar Ch. 1)

II. Predictive DM Techniques
   – Bayesian classifier (Kononenko Ch. 9.6)
   – Decision Tree learning (Mitchell Ch. 3, Kononenko Ch. 9.1)
   – Classification rule learning (Bartholdi book Ch. 7, Kononenko Ch. 9.2)
   – Classifier Evaluation (Bramer Ch. 6)

III. Regression
    (Kononenko Ch. 9.4)

IV. Descriptive DM
   – Predictive vs. descriptive induction
   – Subgroup discovery
   – Association rule learning (Kononenko Ch. 9.3)
   – Hierarchical clustering (Kononenko Ch. 12.3)

V. Relational Data Mining
   – RDM and Inductive Logic Programming (Dzeroski & Lavrač Ch. 5, Ch. 4)
   – Propositionalization approaches
   – Relational subgroup discovery
Part I. Introduction

Data Mining and the KDD process
- DM standards, tools and visualization
- Classification of Data Mining techniques: Predictive and descriptive DM

What is DM

- Extraction of useful information from data: discovering relationships that have not previously been known
- The viewpoint in this course: Data Mining is the application of Machine Learning techniques to solve real-life data analysis problems

Related areas

Database technology and data warehouses
- efficient storage, access and manipulation of data

Statistics, machine learning, pattern recognition and soft computing*
- classification techniques and techniques for knowledge extraction from data
* neural networks, fuzzy logic, genetic algorithms, probabilistic reasoning

Related areas

Text and Web mining
- Web page analysis
- text categorization
- acquisition, filtering and structuring of textual information
- natural language processing

Visualization
- visualization of data and discovered knowledge
**Point of view in this tutorial**

Knowledge discovery using machine learning methods

**Data Mining, ML and Statistics**

- All areas have a long tradition of developing inductive techniques for data analysis.
  - reasoning from properties of a data sample to properties of a population
- DM vs. ML - Viewpoint in this course:
  - Data Mining is the application of Machine Learning techniques to hard real-life data analysis problems
- DM vs. Statistics:
  - Statistics
    - Hypothesis testing when certain theoretical expectations about the data distribution, independence, random sampling, sample size, etc. are satisfied
    - Main approach: best fitting all the available data
  - Data mining
    - Automated construction of understandable patterns, and structured models
    - Main approach: structuring the data space, heuristic search for decision trees, rules, … covering (parts of) the data space

**Data Mining and KDD**

- KDD is defined as “the process of identifying valid, novel, potentially useful and ultimately understandable models/patterns in data.”
- Data Mining (DM) is the key step in the KDD process, performed by using data mining techniques for extracting models or interesting patterns from the data.


**KDD Process**

- KDD process involves several phases:
  - data preparation
  - data mining (machine learning, statistics)
  - evaluation and use of discovered patterns
  - Data mining is the key step, but represents only 15%-25% of the entire KDD process

**MEDIANA – analysis of media research data**

- Questionnaires about journal/magazine reading, watching of TV programs and listening of radio programs, since 1992, about 1200 questions. Yearly publication: frequency of reading/listening/watching, distribution w.r.t. Sex, Age, Education, Buying power...
- Data for 1998, about 8000 questionnaires, covering lifestyle, spare time activities, personal viewpoints, reading/listening/watching of media (yes/no/how much), interest for specific topics in media, social status
- good quality, “clean” data
- table of n-tuples (rows: individuals, columns: attributes, in classification tasks selected class)

**MEDIANA – media research pilot study**

- Patterns uncovering regularities concerning:
  - Which other journals/magazines are read by readers of a particular journal/magazine ?
  - What are the properties of individuals that are consumers of a particular media offer ?
  - Which properties are distinctive for readers of different journals ?
- Induced models: description (association rules, clusters) and classification (decision trees, classification rules)
Simplified association rules
Finding profiles of readers of the Delo daily newspaper
1. read_Marketing_magazine 116 => read_Delo 95 (0.82)
2. read_Financial_News (Finance) 223 => read_Delo 180 (0.81)
3. read_Views (Razgledi) 201 => read_Delo 157 (0.78)
4. read_Money (Denar) 197 => read_Delo 150 (0.76)
5. read_Vip 181 => read_Delo 134 (0.74)
Interpretation: Most readers of Marketing magazine, Financial News, Views, Money and Vip read also Delo.

Simplified association rules (in Slovene)
1. bere_Sara 332 => bere_Slovenske novice 211 (0.64)
2. bere_Ljubezenske zgodbe 283 => bere_Slovenske novice 174 (0.61)
3. bere_Dolenjski list 520 => bere_Slovenske novice 310 (0.6)
4. bere_Omama 154 => bere_Slovenske novice 90 (0.58)
5. bere_Delavska enotnost 177 => bere_Slovenske novice 102 (0.58)
Večina bralcev Sare, Ljubezenskih zgodb, Dolenjskega lista, Omame in Delavske enotnosti bere tudi Slovenske novice.

Simplified association rules (in Slovene)
1. bere_Sportske novosti 303 => bere_Slovenski delnicar 164 (0.54)
2. bere_Sportske novosti 303 => bere_Salomonov oglasnik 155 (0.51)
3. bere_Sportske novosti 303 => bere_Lady 152 (0.5)
Več kot pol bralcev Sportskih novosti bere tudi Slovenskega delničarja, Salomonov oglasnik in Lady.

Decision tree
Finding reader profiles: decision tree for classifying people into readers and non-readers of a teenage magazine.

Part I. Introduction
Data Mining and the KDD process
DM standards, tools and visualization
- Classification of Data Mining techniques: Predictive and descriptive DM

CRISP-DM
- Cross-Industry Standard Process for DM
- A collaborative, 18-months partially EC founded project started in July 1997
- NCR, ISL (Clementine), Daimler-Benz, OHRA (Dutch health insurance companies), and SIG with more than 80 members
- DM from art to engineering
- Views DM more broadly than Fayyad et al. (actually DM is treated as KDD process):
Data Mining and Knowledge Discovery Lecture notes

CRISP Data Mining Process

DM tasks

• Data Preparation
• Data mining
• Model building
• Evaluation
• Deployment

Visualization

• can be used on its own (usually for description and summarization tasks)
• can be used in combination with other DM techniques, for example
  – visualization of decision trees
  – cluster visualization
  – visualization of association rules
  – subgroup visualization

Public DM tools

• WEKA - Waikato Environment for Knowledge Analysis
• Orange
• KNIME - Konstanz Information Miner
• R – Bioconductor, …

Visualization

DB Miner: Association rule visualization

Data visualization:
Scatter plot
Data Mining and Knowledge Discovery
Lecture notes

Part I. Introduction
Data Mining and the KDD process
- DM standards, tools and visualization
- Classification of Data Mining techniques:
  Predictive and descriptive DM

Types of DM tasks
- Predictive DM:
  - Classification (learning of rules, decision trees, ...)
  - Prediction and estimation (regression)
  - Predictive relational DM (ILP)
- Descriptive DM:
  - description and summarization
  - dependency analysis (association rule learning)
  - discovery of properties and constraints
  - segmentation (clustering)
  - subgroup discovery
- Text, Web and image analysis

Predictive vs. descriptive induction
- Predictive induction: Inducing classifiers for solving classification and prediction tasks,
  - Classification rule learning, Decision tree learning, ...
  - Bayesian classifier, ANN, SVM, ...
  - Data analysis through hypothesis generation and testing
- Descriptive induction: Discovering interesting regularities in the data, uncovering patterns, ...
  for solving KDD tasks
  - Symbolic clustering, Association rule learning, Subgroup discovery, ...
  - Exploratory data analysis
Predictive DM formulated as a machine learning task:

- Given a set of labeled training examples (n-tuples of attribute values, labeled by class name)
  
<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>example1</td>
<td>v1,1</td>
<td>v1,2</td>
<td>v1,3</td>
<td>C1</td>
</tr>
<tr>
<td>example2</td>
<td>v2,1</td>
<td>v2,2</td>
<td>v2,3</td>
<td>C2</td>
</tr>
</tbody>
</table>

- By performing generalization from examples (induction) find a hypothesis (classification rules, decision tree, …) which explains the training examples, e.g. rules of the form: 
  
  
  \[ (A_i = v_{i,k}) \land (A_j = v_{j,l}) \land ... \rightarrow \text{Class} = C_n \]

Data Mining in a Nutshell

Given: transaction data table, relational database, text documents, Web pages
Find: a classification model, a set of interesting patterns

Predictive DM - Classification

- data are objects, characterized with attributes - they belong to different classes (discrete labels)
- given objects described with attribute values, induce a model to predict different classes
- decision trees, if-then rules, discriminant analysis, ...

Contact lens data: Decision tree

Type of task: prediction and classification
Hypothesis language: decision trees
(nodes: attributes, arcs: values of attributes, leaves: classes)
**Contact lens data: Classification rules**

**Type of task:** prediction and classification

**Hypothesis language:** rules $X \rightarrow C$, if $X$ then $C$

$X$ conjunction of attribute values, $C$ class

- tear production = reduced $\rightarrow$ lenses = NONE
- tear production = normal & astigmatism = yes & spect. pre. = hypermetrope $\rightarrow$ lenses = NONE
- tear production = normal & astigmatism = no $\rightarrow$ lenses = SOFT
- tear production = normal & astigmatism = yes & spect. pre. = myope $\rightarrow$ lenses = HARD

DEFAULT lenses = NONE

---

**Task reformulation: Concept learning problem**

(positive vs. negative examples of Target class)

<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Spect. presc.</th>
<th>Astigm.</th>
<th>Tear prod.</th>
<th>Lenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>young</td>
<td>myope</td>
<td>no</td>
<td>reduced</td>
<td>NO</td>
</tr>
<tr>
<td>02</td>
<td>young</td>
<td>myope</td>
<td>no</td>
<td>normal</td>
<td>YES</td>
</tr>
<tr>
<td>03</td>
<td>young</td>
<td>myope</td>
<td>yes</td>
<td>reduced</td>
<td>NO</td>
</tr>
<tr>
<td>04</td>
<td>young</td>
<td>myope</td>
<td>yes</td>
<td>normal</td>
<td>YES</td>
</tr>
<tr>
<td>05</td>
<td>young</td>
<td>hypermetrope</td>
<td>no</td>
<td>reduced</td>
<td>NO</td>
</tr>
<tr>
<td>06-013</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>014</td>
<td>pre-presby</td>
<td>hypermetrope</td>
<td>no</td>
<td>normal</td>
<td>YES</td>
</tr>
<tr>
<td>015</td>
<td>pre-presby</td>
<td>hypermetrope</td>
<td>yes</td>
<td>reduced</td>
<td>NO</td>
</tr>
<tr>
<td>016</td>
<td>pre-presby</td>
<td>hypermetrope</td>
<td>yes</td>
<td>normal</td>
<td>NO</td>
</tr>
<tr>
<td>017</td>
<td>pre-presby</td>
<td>myope</td>
<td>no</td>
<td>reduced</td>
<td>NO</td>
</tr>
<tr>
<td>018</td>
<td>pre-presby</td>
<td>myope</td>
<td>no</td>
<td>normal</td>
<td>NO</td>
</tr>
<tr>
<td>019-023</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>024</td>
<td>pre-presby</td>
<td>hypermetrope</td>
<td>yes</td>
<td>normal</td>
<td>NO</td>
</tr>
</tbody>
</table>

---

**Illustrative example: Customer data**

<table>
<thead>
<tr>
<th>Customer</th>
<th>Gender</th>
<th>Age</th>
<th>Income</th>
<th>Spent</th>
<th>BigSpender</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>male</td>
<td>30</td>
<td>214000</td>
<td>18800</td>
<td>yes</td>
</tr>
<tr>
<td>c2</td>
<td>female</td>
<td>19</td>
<td>139000</td>
<td>15100</td>
<td>yes</td>
</tr>
<tr>
<td>c3</td>
<td>male</td>
<td>55</td>
<td>56300</td>
<td>12400</td>
<td>no</td>
</tr>
<tr>
<td>c4</td>
<td>female</td>
<td>48</td>
<td>28000</td>
<td>9800</td>
<td>yes</td>
</tr>
<tr>
<td>c5</td>
<td>male</td>
<td>63</td>
<td>191000</td>
<td>28100</td>
<td>yes</td>
</tr>
<tr>
<td>c6-c13</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>c14</td>
<td>female</td>
<td>61</td>
<td>96500</td>
<td>18100</td>
<td>yes</td>
</tr>
<tr>
<td>c15</td>
<td>male</td>
<td>38</td>
<td>102000</td>
<td>13600</td>
<td>no</td>
</tr>
<tr>
<td>c16</td>
<td>male</td>
<td>38</td>
<td>102000</td>
<td>13600</td>
<td>no</td>
</tr>
<tr>
<td>c17</td>
<td>female</td>
<td>57</td>
<td>215000</td>
<td>29300</td>
<td>yes</td>
</tr>
<tr>
<td>c18</td>
<td>male</td>
<td>33</td>
<td>77000</td>
<td>9700</td>
<td>no</td>
</tr>
<tr>
<td>c19</td>
<td>female</td>
<td>26</td>
<td>95000</td>
<td>11000</td>
<td>no</td>
</tr>
<tr>
<td>c20</td>
<td>female</td>
<td>55</td>
<td>214000</td>
<td>28800</td>
<td>yes</td>
</tr>
</tbody>
</table>

---

**Customer data: Decision trees**

- **Income**
  - $\leq 102000$
  - $> 102000$

- **Age**
  - $\leq 58$
  - $> 58$

- **Gender**
  - = female
  - = male

---

**Customer data: Association rules**

**Type of task:** description (pattern discovery)

**Hypothesis language:** rules $X \rightarrow Y$, if $X$ then $Y$

$X$, $Y$ conjunctions of items (binary-valued attributes)

1. Age $> 52$ & BigSpender = no $\rightarrow$ Sex = male
2. Age $> 52$ & BigSpender = no $\rightarrow$
   Sex = male & Income $\leq 73250$
3. Sex = male & Age $> 52$ & Income $\leq 73250$ $\rightarrow$
   BigSpender = no

---

**Predictive DM - Estimation**

- often referred to as regression
- data are objects, characterized with attributes (discrete or continuous), classes of objects are continuous (numeric)
- given objects described with attribute values, induce a model to predict the numeric class value
- regression trees, linear and logistic regression, ANN, kNN, ...
Customer data: regression tree

\[
\begin{array}{c}
\text{Income} \\
\leq 108000 \\
\text{Age} \leq 42.5 \\
16500 \\
\text{Age} > 42.5 \\
> 108000 \\
12000 \\
\end{array}
\]

In the nodes one usually has
Predicted value ± st. deviation

Relational Data Mining (Inductive Logic Programming) in a Nutshell

Given: a relational database, a set of tables, sets of logical facts, a graph, ...
Find: a classification model, a set of interesting patterns

Relational Data Mining (ILP)

- Learning from multiple tables
- Complex relational problems:
  - temporal data: time series in medicine, traffic control, ...
  - structured data: representation of molecules and their properties in protein engineering, biochemistry, ...
- Illustrative example: structured objects - Trains

Basic table for analysis

<table>
<thead>
<tr>
<th>ID</th>
<th>Zip</th>
<th>Sex</th>
<th>Soc St</th>
<th>Income</th>
<th>Age</th>
<th>Club</th>
<th>Resp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3478</td>
<td>34667</td>
<td>m</td>
<td>si</td>
<td>60-70</td>
<td>32</td>
<td>me</td>
<td>nn</td>
</tr>
<tr>
<td>3479</td>
<td>43666</td>
<td>f</td>
<td>ma</td>
<td>80-90</td>
<td>45</td>
<td>nm</td>
<td>re</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data table presented as logical facts (Prolog format)

```
customer(Id,Zip,Sex,Soc St,Income,Age,Club,Resp)
```

Prolog facts describing data in Table 2:
customer(3478,34667,m,si,60-70,32,me,nn),
customer(3479,43666,f,ma,80-90,45,nm,re).

Expressing a property of a relation:
customer(_,_,f,ma,80-90,45,nm,re).
Relational Data Mining (Inductive Logic Programming)

Data bases:
• Name of relation p
• Attribute of p
• n-tuple \(< v_1, ..., v_n >\) = row in a relational table
• relation p = set of n-tuples = relational table

Logic programming:
• Predicate symbol p
• Argument of predicate p
• Ground fact \(p(v_1, ..., v_n)\)
• Definition of predicate p
• Set of ground facts
• Prolog clause or a set of Prolog clauses

Example predicate definition:
good_customer(C) :-
customer(C,_,female,_,_,_,_,_),
order(C,_,_,_,creditcard).

Part I: Summary

• KDD is the overall process of discovering useful knowledge in data
  – many steps including data preparation, cleaning, transformation, pre-processing
• Data Mining is the data analysis phase in KDD
  – DM takes only 15%-25% of the effort of the overall KDD process
  – employing techniques from machine learning and statistics
• Predictive and descriptive induction have different goals: classifier vs. pattern discovery
• Many application areas
• Many powerful tools available

Part II. Predictive DM techniques

• Naïve Bayesian classifier
• Decision tree learning
• Classification rule learning
• Classifier evaluation

Bayesian methods

• Bayesian methods – simple but powerful classification methods
  – Based on Bayesian formula
  \[ p(H \mid D) = \frac{p(D \mid H) \cdot p(H)}{p(D)} \]
• Main methods:
  – Naïve Bayesian classifier
  – Semi-naïve Bayesian classifier
  – Bayesian networks *

Naïve Bayesian classifier

• Probability of class, for given attribute values

\[ p(c_j \mid v_1, ..., v_n) = \frac{p(v_1, ..., v_n \mid c_j) \cdot p(c_j)}{p(v_1, ..., v_n)} \]

• For all \(C\), compute probability \(p(C_j)\), given values \(v_i\) of all attributes describing the example which we want to classify (assumption: conditional independence of attributes, when estimating \(p(C_j)\) and \(p(C_j \mid v_i)\))

\[ p(c_j \mid v_1, ..., v_n) \approx p(c_j) \cdot \prod_j \frac{p(c_j \mid v_j)}{p(c_j)} \]

• Output \(C_{\text{max}}\) with maximal posterior probability of class:

\[ C_{\text{max}} = \arg \max_{c_i} p(c_i \mid v_1, ..., v_n) \]
Semi-naïve Bayesian classifier

- Naive Bayesian estimation of probabilities (reliable)
  \[ p(c_j | v_i) = \frac{p(c_j)}{p(c_j | v)} \]

- Semi-naïve Bayesian estimation of probabilities (less reliable)
  \[ p(c_j | v, v_k) = \frac{p(c_j | v, v_k)}{p(c_j)} \]

Probability estimation

- Relative frequency:
  \[ p(c_j) = \frac{n(c_j)}{N}, \quad p(c_j | v_i) = \frac{n(c_j, v_i)}{n(v_i)} \] for \( j = 1, \ldots, k \), \( k \) classes

- Prior probability: Laplace law
  \[ p(c_j) = \frac{n(c_j) + 1}{N + k} \]

- m-estimate:
  \[ p(c_j) = \frac{n(c_j) + m \cdot p(c_j)}{N + m} \]

Example of explanation of semi-naïve Bayesian classifier

Hip surgery prognosis

Class = no ("no complications", most probable class, 2 class problem)

<table>
<thead>
<tr>
<th>Attribute value</th>
<th>For decision</th>
<th>Against</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age = 70-80</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>Sex = Female</td>
<td>-0.19</td>
<td></td>
</tr>
<tr>
<td>Mobility before injury = Fully mobile</td>
<td>0.04</td>
<td></td>
</tr>
<tr>
<td>State of health before injury = Other</td>
<td>0.52</td>
<td></td>
</tr>
<tr>
<td>Mechanism of injury = Simple fall</td>
<td>0.08</td>
<td></td>
</tr>
<tr>
<td>Additional injuries = None</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Time between injury and operation &gt; 10 days</td>
<td>0.42</td>
<td></td>
</tr>
<tr>
<td>Fracture classification acc. To Garden = Garden III</td>
<td>-0.3</td>
<td></td>
</tr>
<tr>
<td>Fracture classification acc. To Pauwels = Pauwels III</td>
<td>-0.14</td>
<td></td>
</tr>
<tr>
<td>Transfusion = Yes</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>Antibiotic profilaxies = Yes</td>
<td>-0.32</td>
<td></td>
</tr>
<tr>
<td>Hospital rehabilitation = Yes</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>General complications = None</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information gain</th>
<th>C1</th>
<th>C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v2</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v5</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v6</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>v7</td>
<td>12</td>
<td></td>
</tr>
</tbody>
</table>

Explanation of Bayesian classifier

- Based on information theory
  - Expected number of bits needed to encode a message = optimal code length -log p for a message, whose probability is p (\(^*\))

- Explanation based of the sum of information gains of individual attribute values \( v_i \) (Kononenko and Bratko 1991, Kononenko 1993)
  \[ -\log(p(c_j | v_i, \ldots, v_k)) = -\log(p(c_j)) - \sum_{i=1}^{k} (-\log(p(c_j) + \log(p(c_j | v_i))) \]

\(* \log p denotes binary logarithm \)
Naïve Bayesian classifier

- Naïve Bayesian classifier can be used
  - when we have sufficient number of training examples for reliable probability estimation
- It achieves good classification accuracy
  - can be used as ‘gold standard’ for comparison with other classifiers
- Resistant to noise (errors)
  - Reliable probability estimation
  - Uses all available information
- Successful in many application domains
  - Web page and document classification
  - Medical diagnosis and prognosis, ...

Improved classification accuracy due to using m-estimate

<table>
<thead>
<tr>
<th></th>
<th>Primary tumor</th>
<th>Breast cancer</th>
<th>thyroid</th>
<th>Rheumatology</th>
</tr>
</thead>
<tbody>
<tr>
<td>#instance</td>
<td>339</td>
<td>298</td>
<td>884</td>
<td>355</td>
</tr>
<tr>
<td>#class</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>#attrib</td>
<td>17</td>
<td>10</td>
<td>15</td>
<td>32</td>
</tr>
<tr>
<td>#values</td>
<td>2</td>
<td>2.7</td>
<td>9.1</td>
<td>9.1</td>
</tr>
<tr>
<td>majority</td>
<td>25%</td>
<td>80%</td>
<td>56%</td>
<td>66%</td>
</tr>
<tr>
<td>entropy</td>
<td>3.64</td>
<td>0.72</td>
<td>1.58</td>
<td>1.7</td>
</tr>
</tbody>
</table>

Relative freq. m-estimate

<table>
<thead>
<tr>
<th></th>
<th>Primary tumor</th>
<th>Breast cancer</th>
<th>hepatitis</th>
<th>lymphography</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>48.20%</td>
<td>52.50%</td>
<td>58.40%</td>
<td>79.70%</td>
</tr>
<tr>
<td></td>
<td>77.40%</td>
<td>79.70%</td>
<td>58.40%</td>
<td>90.00%</td>
</tr>
<tr>
<td></td>
<td>79.70%</td>
<td>87.70%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Part II. Predictive DM techniques

- Naïve Bayesian classifier
- Decision tree learning
- Classification rule learning
- Classifier evaluation

Illustrative example: Contact lenses data

Decision tree for contact lenses recommendation

Decision tree for contact lenses recommendation
**PlayTennis: Training examples**

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Decision tree representation for PlayTennis**

- each internal node is a test of an attribute
- each branch corresponds to an attribute value
- each path is a conjunction of attribute values
- each leaf node assigns a classification

**Decision tree representation for PlayTennis**

Decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances:

\[
\begin{align*}
&\text{(Outlook=Sunny \land Humidity=Normal)} \\
&\lor \ (\text{Outlook=Overcast}) \\
&\lor \ (\text{Outlook=Rain \land Wind=Weak})
\end{align*}
\]

**PlayTennis: Other representations**

- Logical expression for PlayTennis=Yes:
  \[(\text{Outlook=Sunny} \land \text{Humidity=Normal}) \lor (\text{Outlook=Overcast}) \lor (\text{Outlook=Rain} \land \text{Wind=Weak})\]

- Converting a tree to if-then rules:
  - IF Outlook=Sunny \land Humidity=Normal THEN PlayTennis=Yes
  - IF Outlook=Overcast THEN PlayTennis=Yes
  - IF Outlook=Rain \land Wind=Weak THEN PlayTennis=Yes
  - IF Outlook=Sunny \land Humidity=Normal THEN PlayTennis=Yes
  - IF Outlook=Rain \land Wind=Strong THEN PlayTennis=No

**Appropriate problems for decision tree learning**

- Classification problems: classify an instance into one of a discrete set of possible categories (medical diagnosis, classifying loan applicants, …)

- Characteristics:
  - instances described by attribute-value pairs (discrete or real-valued attributes)
  - target function has discrete output values (boolean or multi-valued, if real-valued then regression trees)
  - disjunctive hypothesis may be required
  - training data may be noisy (classification errors and/or errors in attribute values)
  - training data may contain missing attribute values
Learning of decision trees

- ID3 (Quinlan 1979), CART (Breiman et al. 1984), C4.5, WEKA, ...
  - create the root node of the tree
  - if all examples from S belong to the same class Cj
    - then label the root with Cj
  - else
    - select the 'most informative' attribute $A$ with values $v_1, v_2, ..., v_n$
    - divide training set $S$ into $S_1, ..., S_n$ according to values $v_1, ..., v_n$
    - recursively build sub-trees $T_1, ..., T_n$ for $S_1, ..., S_n$

Search heuristics in ID3

- Central choice in ID3: Which attribute to test at each node in the tree? The attribute that is most useful for classifying examples.
- Define a statistical property, called information gain, measuring how well a given attribute separates the training examples w.r.t their target classification.
- First define a measure commonly used in information theory, called entropy, to characterize the (im)purity of an arbitrary collection of examples.

Entropy

- $S$ - training set, $C_1, ..., C_N$ - classes
- Entropy $E(S)$ – measure of the impurity of training set $S$
  \[ E(S) = - \sum_{i=1}^{N} p_i \log_2 p_i \]
  - $p_i$ - prior probability of class $C_i$ (relative frequency of $C_i$ in $S$)
- Entropy in binary classification problems
  \[ E(S) = - p_+ \log_2 p_+ - p_- \log_2 p_- \]

Entropy – why?

- Entropy $E(S)$ = expected amount of information (in bits) needed to assign a class to a randomly drawn object in $S$ (under the optimal, shortest-length code)
- Why?
  - Information theory: optimal length code assigns $- \log_2 p$ bits to a message having probability $p$
  - So, in binary classification problems, the expected number of bits to encode + or – of a random member of $S$ is:
    \[ p_+ (- \log_2 p_+) + p_- (- \log_2 p_-) = - p_+ \log_2 p_+ - p_- \log_2 p_- \]

PlayTennis: Entropy

- Training set $S$: 14 examples (9 pos., 5 neg.)
- Notation: $S = [9+, 5-]$
- $E(S) = - p_+ \log_2 p_+ - p_- \log_2 p_-$
- Computing entropy, if probability is estimated by relative frequency
  \[ E(S) = \left( \frac{|S_9|}{|S|} \log_2 \left( \frac{|S_9|}{|S|} \right) \right) + \left( \frac{|S_{5-}|}{|S|} \log_2 \left( \frac{|S_{5-}|}{|S|} \right) \right) \]
- $E([9+, 5-]) = - (9/14) \log_2 (9/14) - (5/14) \log_2 (5/14) = 0.940$
PlayTennis: Entropy

- \( E(S) = - p_+ \log_2 p_+ - p_- \log_2 p_- \)
- \( E(9+,5-) = -(9/14) \log_2(9/14) - (5/14) \log_2(5/14) = 0.940 \)

Information gain search heuristic

- Information gain measure is aimed to minimize the number of tests needed for the classification of a new object
- \( \text{Gain}(S,A) = \text{expected reduction in entropy of } S \text{ due to sorting on } A \)
- \( \text{Gain}(S,A) = E(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} E(S_v) \)
- Most informative attribute: \( \max \text{Gain}(S,A) \)

Information gain search heuristic

- Which attribute is more informative, \( A_1 \) or \( A_2 \) ?
- \( \text{Gain}(S,A_1) = 0.94 - (8/14 \times 0.811 + 6/14 \times 1.00) = 0.048 \)
- \( \text{Gain}(S,A_2) = 0.94 - 0 = 0.94 \) \( A_2 \) has max Gain

PlayTennis: Information gain

- Which attribute is the best?
  - \( \text{Gain}(S,\text{Outlook})=0.246 \) MAX !
  - \( \text{Gain}(S,\text{Humidity})=0.151 \)
  - \( \text{Gain}(S,\text{Wind})=0.048 \)
  - \( \text{Gain}(S,\text{Temperature})=0.029 \)

PlayTennis: Information gain

- Which attribute should be tested here?
  - \( \text{Gain}(S_{\text{sunny}}, \text{Humidity}) = 0.97-(3/5)0-(2/5)0 = 0.970 \) MAX !
  - \( \text{Gain}(S_{\text{sunny}}, \text{Temperature}) = 0.97-(2/5)0-(2/5)1-(1/5)0 = 0.570 \)
  - \( \text{Gain}(S_{\text{sunny}}, \text{Wind}) = 0.97-(2/5)1-(3/5)0.918 = 0.019 \)
Probability estimates

- **Relative frequency**: problems with small samples
  \[ p(\text{Class}\mid\text{Cond}) = \frac{n(\text{Class}\mid\text{Cond})}{n(\text{Cond})} \]
  
  \[
  \begin{align*}
  [6+,1-] (7) &= 6/7 \\
  [2+,0-] (2) &= 2/2 = 1
  \end{align*}
  \]

- **Laplace estimate**: assumes uniform prior distribution of k classes
  \[ p(\text{Class}\mid\text{Cond}) = \frac{n(\text{Class}\mid\text{Cond}) + 1}{n(\text{Cond}) + k} \]
  
  \[
  \begin{align*}
  [6+,1-] (7) &= \frac{6+1}{7+2} = 7/9 \\
  [2+,0-] (2) &= \frac{2+1}{2+2} = 3/4
  \end{align*}
  \]

Heuristic search in ID3

- **Search bias**: Search the space of decision trees from simplest to increasingly complex (greedy search, no backtracking, prefer small trees)
- **Search heuristics**: At a node, select the attribute that is most useful for classifying examples, split the node accordingly
- **Stopping criteria**: A node becomes a leaf
  - if all examples belong to same class \( C_j \), label the leaf with \( C_j \)
  - if all attributes were used, label the leaf with the most common value \( C_k \) of examples in the node
- **Extension to ID3**: handling noise - tree pruning

Pruning of decision trees

- Avoid overfitting the data by tree pruning
- Pruned trees are
  - less accurate on training data
  - more accurate when classifying unseen data

Handling noise – Tree pruning

Sources of imperfection
1. Random errors (noise) in training examples
   - erroneous attribute values
   - erroneous classification
2. Too sparse training examples (incompleteness)
3. Inappropriate/insufficient set of attributes (inexactness)
4. Missing attribute values in training examples

Prediction of breast cancer recurrence: Tree pruning

<table>
<thead>
<tr>
<th>Degree of malignancy</th>
<th>Tumor size</th>
<th>Involved nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 3</td>
<td>≥ 3</td>
<td>≥ 3</td>
</tr>
<tr>
<td>≥ 15</td>
<td>&lt; 15</td>
<td>&lt; 3</td>
</tr>
<tr>
<td>≥ 15</td>
<td>&gt; 15</td>
<td>≥ 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age</th>
<th>no_recur 125</th>
<th>no_recur 30</th>
<th>no_recur 27</th>
<th>no_recur 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 40</td>
<td>recurrence 39</td>
<td>recurrence 18</td>
<td>recurrence 10</td>
<td>recurrence 1</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
</tr>
<tr>
<td></td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
</tr>
<tr>
<td></td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
<td>no_recur 4</td>
</tr>
</tbody>
</table>

Handling noise – Tree pruning

- Handling imperfect data
  - handling imperfections of type 1-3
    - pre-pruning (stopping criteria)
    - post-pruning / rule truncation
  - handling missing values
- Pruning avoids perfectly fitting noisy data: relaxing the completeness (fitting all +) and consistency (fitting all -) criteria in ID3
Accuracy and error

• Accuracy: percentage of correct classifications
  – on the training set
  – on unseen instances
• How accurate is a decision tree when classifying unseen instances
  – An estimate of accuracy on unseen instances can be computed, e.g., by averaging over 4 runs:
    • split the example set into training set (e.g. 70%) and test set (e.g. 30%)
    • induce a decision tree from training set, compute its accuracy on test set
• Error = 1 - Accuracy
• High error may indicate data overfitting

Overfitting and accuracy

• Typical relation between tree size and accuracy

How to select the “best” tree

• Measure performance over training data (e.g., pessimistic post-pruning, Quinlan 1993)
• Measure performance over separate validation data set (e.g., reduced error pruning, Quinlan 1987)
  – until further pruning is harmful DO:
    • for each node evaluate the impact of replacing a subtree by a leaf, assigning the majority class of examples in the leaf, if the pruned tree performs no worse than the original over the validation set
    • greedily select the node whose removal most improves tree accuracy over the validation set
• MDL: minimize size(tree) + size(misclassifications(tree))

Avoiding overfitting

• How can we avoid overfitting?
  – Pre-pruning (forward pruning): stop growing the tree e.g., when data split not statistically significant or too few examples are in a split
  – Post-pruning: grow full tree, then post-prune

• forward pruning considered inferior (myopic)
• post-pruning makes use of sub trees

Selected decision/regression tree learners

• Decision tree learners
  – ID3 (Quinlan 1979)
  – CART (Breiman et al. 1984)
  – Assistant (Cestnik et al. 1987)
  – C4.5 (Quinlan 1993), C5 (See5, Quinlan)
  – J48 (available in WEKA)
• Regression tree learners, model tree learners
  – M5, M5P (implemented in WEKA)

Features of C4.5

• Implemented as part of the WEKA data mining workbench
• Handling noisy data: post-pruning
• Handling incompletely specified training instances: ‘unknown’ values (?)
  – in learning assign conditional probability of value v: \( p(v|C) = p(v|C) / p(C) \)
  – in classification: follow all branches, weighted by prior prob. of missing attribute values
Other features of C4.5

- Binarization of attribute values
  - for continuous values select a boundary value maximally increasing the informativity of the attribute: sort the values and try every possible split (done automatically)
  - for discrete values try grouping the values until two groups remain
- 'Majority' classification in NULL leaf (with no corresponding training example)
  - if an example 'falls' into a NULL leaf during classification, the class assigned to this example is the majority class of the parent of the NULL leaf

*the basic C4.5 doesn't support binarization of discrete attributes, it supports grouping

Part II. Predictive DM techniques

- Naïve Bayesian classifier
- Decision tree learning
- Classification rule learning
- Classifier evaluation

Rule Learning in a Nutshell

Given: transaction data table, relational database (a set of objects, described by attribute values)
Find: a classification model in the form of a set of rules; or a set of interesting patterns in the form of individual rules

Rule set representation

- Rule base is a disjunctive set of conjunctive rules
- Standard form of rules:
  IF Condition THEN Class
  Class IF Conditions
  Class ← Conditions

- Form of CN2 rules:
  IF Conditions THEN MajClass [ClassDistr]

- Rule base: \{R1, R2, R3, ..., DefaultRule\}

Data mining example

Input: Contact lens data

<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Spect. pre.</th>
<th>Astigm.</th>
<th>Tear prod.</th>
<th>Lenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>O1</td>
<td>young</td>
<td>myope</td>
<td>no</td>
<td>reduced</td>
<td>NONE</td>
</tr>
<tr>
<td>O2</td>
<td>young</td>
<td>myope</td>
<td>no</td>
<td>normal</td>
<td>SOFT</td>
</tr>
<tr>
<td>O3</td>
<td>young</td>
<td>myope</td>
<td>yes</td>
<td>reduced</td>
<td>NONE</td>
</tr>
<tr>
<td>O4</td>
<td>young</td>
<td>myope</td>
<td>yes</td>
<td>normal</td>
<td>HARD</td>
</tr>
<tr>
<td>O5</td>
<td>young</td>
<td>hypermetrope</td>
<td>no</td>
<td>reduced</td>
<td>NONE</td>
</tr>
<tr>
<td>O6-O13</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>O14</td>
<td>presbyopic</td>
<td>hypermetrope</td>
<td>no</td>
<td>normal</td>
<td>SOFT</td>
</tr>
<tr>
<td>O15</td>
<td>presbyopic</td>
<td>hypermetrope</td>
<td>yes</td>
<td>reduced</td>
<td>NONE</td>
</tr>
<tr>
<td>O16</td>
<td>presbyopic</td>
<td>hypermetrope</td>
<td>yes</td>
<td>normal</td>
<td>NONE</td>
</tr>
<tr>
<td>O17</td>
<td>presbyopic</td>
<td>myope</td>
<td>no</td>
<td>reduced</td>
<td>NONE</td>
</tr>
<tr>
<td>O18</td>
<td>presbyopic</td>
<td>myope</td>
<td>no</td>
<td>normal</td>
<td>NONE</td>
</tr>
<tr>
<td>O19-O23</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>O24</td>
<td>presbyopic</td>
<td>hypermetrope</td>
<td>yes</td>
<td>normal</td>
<td>NONE</td>
</tr>
</tbody>
</table>

Contact lens data: Classification rules

Type of task: prediction and classification
Hypothesis language: rules \(X \rightarrow C\), if \(X\) then \(C\)
\(X\) conjunction of attribute values, \(C\) class

tear production=reduced \(\rightarrow\) lenses=NONE
tear production=normal \& astigmatism=yes \& spect. pre.=hypermetrope \(\rightarrow\) lenses=NONE
tear production=normal \& astigmatism=no \(\rightarrow\) lenses=SOFT
tear production=normal \& astigmatism=yes \& spect. pre.=myope \(\rightarrow\) lenses=HARD
DEFAULT lenses=NONE
Rule learning

- Two rule learning approaches:
  - Learn decision tree, convert to rules
  - Learn set/list of rules
    - Learning an unordered set of rules
    - Learning an ordered list of rules
- Heuristics, overfitting, pruning

Contact lenses: convert decision tree to an unordered rule set

Contact lenses: convert decision tree to decision list

Converting decision tree to rules, and rule post-pruning (Quinlan 1993)

- Very frequently used method, e.g., in C4.5 and J48
- Procedure:
  - grow a full tree (allowing overfitting)
  - convert the tree to an equivalent set of rules
  - prune each rule independently of others
  - sort final rules into a desired sequence for use

Concept learning: Task reformulation for rule learning: (pos. vs. neg. examples of Target class)

Original covering algorithm (AQ, Michalski 1969,86)

Given examples of N classes C_1, ..., C_N for each class C_i do
- E_i := P_i U N_i (P_i pos., N_i neg.)
- RuleBase(C_i) := empty
- repeat (learn-set-of-rules)
  - learn-one-rule R covering some positive examples and no negatives
  - add R to RuleBase(C_i)
  - delete from P_i all pos. ex. covered by R
  - until P_i = empty
Covering algorithm

Positive examples

Negative examples

Rule1: \( C_l =+ \) \( \leftarrow \) Cond2 AND Cond3

Rule2: \( C_l =+ \) \( \leftarrow \) Cond8 AND Cond6

PlayTennis: Training examples

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Slight</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Slight</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Slight</td>
<td>No</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Heuristics for learn-one-rule: PlayTennis example

PlayTennis = yes [9+,5−] (14)

PlayTennis = yes ← Wind=weak [6+,2−] (8)
← Wind=strong [3+,3−] (6)
← Humidity=normal [6+,1−] (7)
← ...

PlayTennis = yes ← Humidity=normal
← Outlook=sunny [2+,0−] (2)
← ...

Estimating rule accuracy (rule precision) with the probability that a covered example is positive:

A(Class ← Cond) = \( p(\text{Class}|\text{Cond}) \)

Estimating the probability with the relative frequency of covered pos. ex. / all covered ex.:

\[
\begin{align*}
[6+,1−] (7) &= \frac{6}{7} \\
[2+,0−] (2) &= \frac{2}{2} = 1
\end{align*}
\]

Learn-one-rule: search heuristics

- Assume a two-class problem
- Two classes (+,-), learn rules for + class (Cl).
- Search for specializations R' of a rule R = Cl ← Cond from the RuleBase.
- Specialization R' of rule R = Cl ← Cond has the form R' = Cl ← Cond & Cond'
- Heuristic search for rules: find the 'best' Cond to be added to the current rule R, such that rule accuracy is improved, e.g., such that Acc(R') > Acc(R)
  - where the expected classification accuracy can be estimated as A(R) = \( p(\text{Cl}|\text{Cond}) \)

Learn-one-rule as search: PlayTennis example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis = yes</td>
<td>[9+,5−] (14)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=weak [6+,2−] (8)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=strong [3+,3−] (6)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal [6+,1−] (7)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=weak</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=strong</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=sunny</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=rain</td>
</tr>
</tbody>
</table>

Learn-one-rule as heuristic search: PlayTennis example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis = yes</td>
<td>[9+,5−] (14)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=weak [6+,2−] (8)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=strong [3+,3−] (6)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal [6+,1−] (7)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=weak</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=strong</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=sunny</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=rain</td>
</tr>
</tbody>
</table>

Learn-one-rule: Greedy vs. beam search

- learn-one-rule by greedy general-to-specific search, at each step selecting the 'best' descendant, no backtracking
  - e.g., the best descendant of the initial rule
    - PlayTennis = yes ← Wind=weak
  - is rule PlayTennis = yes ← Humidity=normal
- beam search: maintain a list of k best candidates at each step; descendants (specializations) of each of these k candidates are generated, and the resulting set is again reduced to k best candidates

Probability estimates

- Relative frequency:
  - problems with small samples
  - \( \frac{p(Class|Cond)}{n(Class,Cond) + 1} \cdot \frac{n(Cond) + k}{k} \)
    - \([6+,1−] (7) = 6+1 / 7+2 = 7/9 \)
    - \([2+,0−] (2) = 2+1 / 2+2 = 3/4 \)

Learn-one-rule as heuristic search: PlayTennis example

<table>
<thead>
<tr>
<th>Rule</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis = yes</td>
<td>[9+,5−] (14)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=weak [6+,2−] (8)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Wind=strong [3+,3−] (6)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal [6+,1−] (7)</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=weak</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Wind=strong</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=sunny</td>
</tr>
<tr>
<td>PlayTennis = yes</td>
<td>IF Humidity=normal, Outlook=rain</td>
</tr>
</tbody>
</table>
What is “high” rule accuracy (rule precision)?

- Rule evaluation measures:
  - aimed at maximizing classification accuracy
  - minimizing Error = 1 - Accuracy
  - avoiding overfitting
- BUT: Rule accuracy/precision should be traded off against the “default” accuracy/precision of the rule \( Cl \leftarrow \text{true} \)
  - 68% accuracy is OK if there are 20% examples of that class in the training set, but bad if there are 80%
- Relative accuracy
  - \( \text{RAcc}(Cl \leftarrow \text{Cond}) = p(Cl | \text{Cond}) - p(Cl) \)

Weighted relative accuracy

- If a rule covers a single example, its accuracy/precision is either 0% or 100%
  - maximising relative accuracy tends to produce many overly specific rules
- Weighted relative accuracy
  \( \text{WRAcc}(Cl \leftarrow \text{Cond}) = p(\text{Cond}) \cdot [p(Cl | \text{Cond}) - p(Cl)] \)
  - \( \text{WRAcc} \) is a fundamental rule evaluation measure:
    - \( \text{WRAcc} \) can be used if you want to assess both accuracy and significance
    - \( \text{WRAcc} \) can be used if you want to compare rules with different heads and bodies

Learn-one-rule: search heuristics

- Assume two classes (+,-), learn rules for + class (Cl). Search for specializations of one rule \( R = Cl \leftarrow \text{Cond} \) from RuleBase.
- Expected classification accuracy: \( A(R) = p(Cl|\text{Cond}) \)
- Informativity (info needed to specify that example covered by \( \text{Cond} \) belongs to \( Cl \)): \( I(R) = - \log_2 p(Cl|\text{Cond}) \)
- Accuracy gain (increase in expected accuracy):
  \( \text{AG}(R',R) = p(Cl|\text{Cond'}) - p(Cl|\text{Cond}) \)
- Information gain (decrease in the information needed):
  \( \text{IG}(R',R) = \log_2 p(Cl|\text{Cond}) - \log_2 p(Cl|\text{Cond'}) \)
- Weighted measures favoring more general rules: \( \text{WAG}, \text{WIG} \)
  \( \text{WAG}(R',R) = p(\text{Cond'})/p(\text{Cond}) \cdot (p(Cl|\text{Cond'}) - p(Cl|\text{Cond})) \)
- Weighted relative accuracy trades off coverage and relative accuracy
  \( \text{WRRAcc}(R) = p(\text{Cond}).(p(Cl|\text{Cond}) - p(Cl)) \)

Ordered set of rules: if-then-else rules

- rule Class IF Conditions is learned by first determining Conditions and then Class
- Notice: mixed sequence of classes \( C_1, \ldots, C_n \) in RuleBase
- But: ordered execution when classifying a new instance: rules are sequentially tried and the first rule that ‘fires’ (covers the example) is used for classification
- Decision list \( \{R_1, R_2, R_3, \ldots, D\} \): rules \( R_i \) are interpreted as if-then-else rules
  - If no rule fires, then DefaultClass (majority class in \( E_{\text{cur}} \))

Sequential covering algorithm (similar as in Mitchell’s book)

- RuleBase := empty
- \( E_{\text{cur}} := E \)
- repeat
  - learn-one-rule \( R \)
  - RuleBase := RuleBase U \( R \)
  - \( E_{\text{cur}} := E_{\text{cur}} - \{\text{examples covered and correctly classified by } R\} \) (DELETE ONLY POS. EX.)
  - until performance(\( R, E_{\text{cur}} \) < ThresholdR
- RuleBase := sort RuleBase by performance(\( R,E \))
- return RuleBase

Learn ordered set of rules (CN2, Clark and Niblett 1989)

- RuleBase := empty
- \( E_{\text{cur}} := E \)
- repeat
  - learn-one-rule \( R \)
  - RuleBase := RuleBase U \( R \)
  - \( E_{\text{cur}} := E_{\text{cur}} - \{\text{all examples covered by } R\} \) (NOT ONLY POS. EX.)
  - until performance(\( R, E_{\text{cur}} \) < ThresholdR
- RuleBase := sort RuleBase by performance(\( R,E \))
- RuleBase := RuleBase U DefaultRule(\( E_{\text{cur}} \))
Learn-one-rule: Beam search in CN2

- Beam search in CN2 learn-one-rule algo.:
  - construct BeamSize of best rule bodies (conjunctive conditions) that are statistically significant
  - BestBody - min. entropy of examples covered by Body
  - construct best rule \( R := \text{Head} - \text{BestBody} \) by adding majority class of examples covered by BestBody in rule Head
- performance \( (R, E_{\text{cur}}) : -\text{Entropy}(E_{\text{cur}}) \):
  - performance\( (R, E_{\text{cur}}) < \text{Threshold}_R \) (neg. num.)
  - Why? Ent. > \( t \) is bad, Perf. = \(-\text{Ent} < -t\) is bad

Variations

- Sequential vs. simultaneous covering of data (as in TDIDT): choosing between attribute-values vs. choosing attributes
- Learning rules vs. learning decision trees and converting them to rules
- Pre-pruning vs. post-pruning of rules
- What statistical evaluation functions to use
- Probabilistic classification

Probabilistic classification

- In the ordered case of standard CN2 rules are interpreted in an IF-THEN-ELSE fashion, and the first fired rule assigns the class.
- In the unordered case all rules are tried and all rules which fire are collected. If a clash occurs, a probabilistic method is used to resolve the clash.
- A simplified example:
  1. tear production=reduced \(\Rightarrow\) lenses=NONE \([S=0,H=0,N=12]\)
  2. tear production=normal \& astigmatism=yes \& spect. pre.=hypermetrope \(\Rightarrow\) lenses=NONE \([S=0,H=1,N=2]\)
  3. tear production=normal \& astigmatism=no \(\Rightarrow\) lenses=SOFT \([S=5,H=0,N=1]\)
  4. tear production=normal \& astigmatism=yes \& spect. pre.=myope \(\Rightarrow\) lenses=HARD \([S=0,H=3,N=2]\)
  5. DEFAULT lenses=NONE

Suppose we want to classify a person with normal tear production and astigmatism. Two rules fire: rule 2 with coverage \([S=0,H=1,N=2]\) and rule 4 with coverage \([S=0,H=3,N=2]\). The classifier computes total coverage as \([S=5,H=4,N=4]\), resulting in probabilistic classification into class \(H\) with probability 0.5 and \(N\) with probability 0.5. In this case, the clash can not be resolved, as both probabilities are equal.

Part II. Predictive DM techniques

- Naïve Bayesian classifier
- Decision tree learning
- Classification rule learning
- Classifier evaluation

Classifier evaluation

- Accuracy and Error
- n-fold cross-validation
- Confusion matrix
- ROC

Evaluating hypotheses

- Use of induced hypotheses
  - discovery of new patterns, new knowledge
  - classification of new objects
- Evaluating the quality of induced hypotheses
  - Accuracy, Error = 1 - Accuracy
  - classification accuracy on testing examples = percentage of correctly classified instances
    - split the example set into training set (e.g. 70%) to induce a concept, and test set (e.g. 30%) to test its accuracy
    - more elaborate strategies: 10-fold cross validation, leave-one-out, ...
  - comprehensibility (compactness)
  - information contents (information score), significance
**n-fold cross validation**

- A method for accuracy estimation of classifiers
- Partition set $D$ into $n$ disjoint, almost equally-sized folds $T_i$ where $U_i T_i = D$
- for $i = 1, ..., n$
  - form a training set out of $n-1$ folds: $D_i = D \setminus T_i$
  - induce classifier $H_i$ from examples in $D_i$
  - use fold $T_i$ for testing the accuracy of $H_i$
- Estimate the accuracy of the classifier by averaging accuracies over 10 folds $T_i$

**Confusion matrix and rule (in)accuracy**

- Accuracy of a classifier is measured as $\frac{TP+TN}{N}$.
- Suppose two rules are both 80% accurate on an evaluation dataset, are they always equally good?  
  - e.g., Rule 1 correctly classifies 40 out of 50 positives and 40 out of 50 negatives; Rule 2 correctly classifies 30 out of 50 positives and 50 out of 50 negatives  
  - on a test set which has more negatives than positives, Rule 2 is preferable;  
  - on a test set which has more positives than negatives, Rule 1 is preferable; unless…  
  - …the proportion of positives becomes so high that the ‘always positive’ predictor becomes superior!  
- Conclusion: classification accuracy is not always an appropriate rule quality measure
Confusion matrix

- also called *contingency table*

### Classifier 1

<table>
<thead>
<tr>
<th>Predicted positive</th>
<th>Predicted negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive examples</td>
<td>False negatives</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>Negative examples</td>
<td>False positives</td>
</tr>
<tr>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>

### Classifier 2

<table>
<thead>
<tr>
<th>Predicted positive</th>
<th>Predicted negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive examples</td>
<td>False negatives</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>Negative examples</td>
<td>False positives</td>
</tr>
<tr>
<td>0</td>
<td>50</td>
</tr>
</tbody>
</table>

#### ROC space

- **True positive rate** = \( \frac{\text{true pos.}}{\text{pos.}} \)
  - \( \text{TPR}_1 = \frac{40}{50} = 80\% \)
  - \( \text{TPR}_2 = \frac{30}{50} = 60\% \)
- **False positive rate** = \( \frac{\text{false pos.}}{\text{neg.}} \)
  - \( \text{FPR}_1 = \frac{10}{50} = 20\% \)
  - \( \text{FPR}_2 = \frac{0}{50} = 0\% \)
- **ROC space** has
  - FPR on X axis
  - TPR on Y axis

The ROC space

The ROC convex hull

Summary of evaluation

- 10-fold cross-validation is a standard classifier evaluation method used in machine learning
- ROC analysis is very natural for rule learning and subgroup discovery
  - can take costs into account
  - here used for evaluation
  - also possible to use as search heuristic

Part III. Numeric prediction

- Baseline
- Linear Regression
- Regression tree
- Model Tree
- kNN
**Data Mining and Knowledge Discovery**

**Lecture notes**

<table>
<thead>
<tr>
<th>Regression</th>
<th>Classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data: attribute-value description</td>
<td>Target variable: Categorical (nominal)</td>
</tr>
<tr>
<td>Target variable: Continuous</td>
<td>Error: MSE, MAE, RMSE, ...</td>
</tr>
<tr>
<td>Evaluation: cross validation, separate test set, ...</td>
<td>Error: 1-accuracy</td>
</tr>
<tr>
<td>Algorithms: Linear regression, regression trees, ...</td>
<td>Algorithms: Decision trees, Naïve Bayes, ...</td>
</tr>
<tr>
<td>Baseline predictor: Mean of the target variable</td>
<td>Baseline predictor: Majority class</td>
</tr>
</tbody>
</table>

**Example**

- data about 80 people: Age and Height

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1.03</td>
</tr>
<tr>
<td>5</td>
<td>1.19</td>
</tr>
<tr>
<td>6</td>
<td>1.26</td>
</tr>
<tr>
<td>9</td>
<td>1.30</td>
</tr>
<tr>
<td>15</td>
<td>1.60</td>
</tr>
<tr>
<td>19</td>
<td>1.67</td>
</tr>
<tr>
<td>22</td>
<td>1.86</td>
</tr>
<tr>
<td>25</td>
<td>1.85</td>
</tr>
<tr>
<td>41</td>
<td>1.59</td>
</tr>
<tr>
<td>48</td>
<td>1.60</td>
</tr>
<tr>
<td>54</td>
<td>1.90</td>
</tr>
<tr>
<td>71</td>
<td>1.82</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Test set**

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
</tr>
</tbody>
</table>

**Baseline numeric predictor**

- Average of the target variable

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Average predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
<td>1.63</td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

**Baseline predictor: prediction**

Average of the target variable is 1.63

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

**Linear Regression Model**

\[ \text{Height} = 0.0056 \times \text{Age} + 1.4181 \]
Linear Regression: prediction

Height = 0.0056 * Age + 1.4181

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Linear regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
<td></td>
</tr>
</tbody>
</table>

Regression tree

Regression tree: prediction

Model tree

Model tree: prediction

kNN – K nearest neighbors

• Looks at K closest examples (by age) and predicts the average of their target variable
• K=3
Data Mining and Knowledge Discovery
Lecture notes

**kNN prediction**

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.90</td>
</tr>
<tr>
<td>2</td>
<td>1.01</td>
</tr>
<tr>
<td>3</td>
<td>1.03</td>
</tr>
<tr>
<td>4</td>
<td>1.07</td>
</tr>
<tr>
<td>5</td>
<td>1.19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1.36</td>
</tr>
<tr>
<td>9</td>
<td>1.45</td>
</tr>
<tr>
<td>10</td>
<td>1.33</td>
</tr>
<tr>
<td>11</td>
<td>1.49</td>
</tr>
<tr>
<td>12</td>
<td>1.66</td>
</tr>
<tr>
<td>13</td>
<td>1.59</td>
</tr>
<tr>
<td>14</td>
<td>1.58</td>
</tr>
</tbody>
</table>

**Which predictor is the best?**

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Baseline</th>
<th>Linear regression</th>
<th>Regression tree</th>
<th>Model tree</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
<td>1.63</td>
<td>1.43</td>
<td>1.39</td>
<td>1.20</td>
<td>1.01</td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
<td>1.63</td>
<td>1.47</td>
<td>1.46</td>
<td>1.47</td>
<td>1.51</td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
<td>1.63</td>
<td>1.61</td>
<td>1.71</td>
<td>1.71</td>
<td>1.67</td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
<td>1.63</td>
<td>1.81</td>
<td>1.71</td>
<td>1.75</td>
<td>1.81</td>
</tr>
</tbody>
</table>

**Evaluating numeric prediction**

<table>
<thead>
<tr>
<th>Age</th>
<th>Height</th>
<th>Baseline</th>
<th>Linear regression</th>
<th>Regression tree</th>
<th>Model tree</th>
<th>kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.85</td>
<td>1.63</td>
<td>1.43</td>
<td>1.39</td>
<td>1.20</td>
<td>1.01</td>
</tr>
<tr>
<td>10</td>
<td>1.4</td>
<td>1.63</td>
<td>1.47</td>
<td>1.46</td>
<td>1.47</td>
<td>1.51</td>
</tr>
<tr>
<td>35</td>
<td>1.7</td>
<td>1.63</td>
<td>1.61</td>
<td>1.71</td>
<td>1.71</td>
<td>1.67</td>
</tr>
<tr>
<td>70</td>
<td>1.6</td>
<td>1.63</td>
<td>1.81</td>
<td>1.71</td>
<td>1.75</td>
<td>1.81</td>
</tr>
</tbody>
</table>
Part IV. Descriptive DM techniques

- Predictive vs. descriptive induction
- Subgroup discovery
- Association rule learning
- Hierarchical clustering

Predictive vs. descriptive induction

- **Predictive induction**: Inducing classifiers for solving classification and prediction tasks,
  - Classification rule learning, Decision tree learning, ...
  - Bayesian classifier, ANN, SVM, ...
  - Data analysis through hypothesis generation and testing
- **Descriptive induction**: Discovering interesting regularities in the data, uncovering patterns, ... for solving KDD tasks
  - Symbolic clustering, Association rule learning, Subgroup discovery, ...
  - Exploratory data analysis

Descriptive DM

- Often used for preliminary explanatory data analysis
- User gets feel for the data and its structure
- Aims at deriving descriptions of characteristics of the data
- Visualization and descriptive statistical techniques can be used

Supervised vs. unsupervised learning: A rule learning perspective

- **Supervised learning**: Rules are induced from labeled instances (training examples with class assignment) - usually used in **predictive induction**
- **Unsupervised learning**: Rules are induced from unlabeled instances (training examples with no class assignment) - usually used in **descriptive induction**
- Exception: Subgroup discovery
  Discovering individual rules describing interesting regularities in the data from labeled examples
Part IV. Descriptive DM techniques

• Predictive vs. descriptive induction
• Subgroup discovery
• Association rule learning
• Hierarchical clustering

Subgroup Discovery

**Given:** a population of individuals and a target class label (the property of individuals we are interested in)

**Find:** population subgroups that are statistically most ‘interesting’, e.g., are as large as possible and have most unusual statistical (distributional) characteristics w.r.t. the target class (property of interest)

Subgroup interestingness

**Interestingness criteria:**

– As large as possible
– Class distribution as different as possible from the distribution in the entire data set
– Significant
– Surprising to the user
– Non-redundant
– Simple
– Useful - actionable

Subgroup Discovery: Medical Case Study

• Find and characterize population subgroups with high risk for coronary heart disease (CHD) (Gamberger, Lavrač, Krstačič)

• A1 for males: **principal risk factors**
  
  CHD ← positive fam. history & age > 46

• A2 for females: **principal risk factors**
  
  CHD ← bodyMassIndex > 25 & age > 63

• A1, A2 (anamnestic info only), B1, B2 (an. and physical examination), C1 (an., phy. and ECG)

• A1: **supporting factors** (found by statistical analysis):
  
  psychosocial stress, as well as cigarette smoking, hypertension and overweight

Subgroup visualization

Subgroups vs. classifiers

• Classifiers:
  
  – Classification rules aim at pure subgroups
  – A set of rules forms a domain model

• Subgroups:
  
  – Rules describing subgroups aim at significantly higher proportion of positives
  – Each rule is an independent chunk of knowledge

• Link:
  
  – SD can be viewed as cost-sensitive classification
  – Instead of FNcost we aim at increased TPprofit
**Classification Rule Learning for Subgroup Discovery: Deficiencies**

- Only first few rules induced by the covering algorithm have sufficient support (coverage)
- Subsequent rules are induced from smaller and strongly biased example subsets (pos. examples not covered by previously induced rules), which hinders their ability to detect population subgroups
- ‘Ordered’ rules are induced and interpreted sequentially as a if-then-else decision list

**CN2-SD: Adapting CN2 Rule Learning to Subgroup Discovery**

- Weighted covering algorithm
- Weighted relative accuracy (WRAcc) search heuristics, with added example weights
- Probabilistic classification
- Evaluation with different interestingness measures

**CN2-SD: CN2 Adaptations**

- General-to-specific search (beam search) for best rules
- Rule quality measure:
  - CN2: Laplace: \( \text{Acc}(\text{Class} \leftarrow \text{Cond}) = \frac{p(\text{Class}|\text{Cond})}{n_{\text{rule}+k}} \)
  - CN2-SD: Weighted Relative Accuracy
    \( \text{WRAcc}(\text{Class} \leftarrow \text{Cond}) = \frac{p(\text{Cond})}{p(\text{Class}|\text{Cond}) - p(\text{Class})} \)
- Weighted covering approach (example weights)
- Significance testing (likelihood ratio statistics)
- Output: Unordered rule sets (probabilistic classification)

**CN2-SD: Weighted Covering**

- Standard covering approach: covered examples are deleted from current training set
- Weighted covering approach:
  - weights assigned to examples
  - covered pos. examples are re-weighted:
    in all covering loop iterations, store count how many times (with how many rules induced so far) a pos. example has been covered: \( w(e,i), w(e,0)=1 \)
    - Additive weights: \( w(e,i) = 1/(i+1) \)
    - \( w(e,i) \) = pos. example \( e \) being covered \( i \) times

**Subgroup Discovery**

Positive examples

Negative examples

Positive examples

Rule1: \( \text{Cl}++ \leftarrow \text{Cond6 AND Cond2} \)

Positive examples

Negative examples
Subgroup Discovery

Positive examples

Negative examples

Rule 2: \( C_1 \leftarrow Cond3 \text{ AND } Cond4 \)

Subgroup Discovery

Positive examples

Negative examples

CN2-SD: Weighted WRAcc Search Heuristic

- Weighted relative accuracy (WRAcc) search heuristics, with added example weights
  
  \[
  \text{WRAcc}(C1 \leftarrow Cond) = p(Cond) (p(Cl|Cond) - p(Cl))
  \]
  
  increased coverage, decreased # of rules, approx. equal accuracy (PKDD-2000)

- In WRAcc computation, probabilities are estimated with relative frequencies, adapt:
  
  \[
  \text{WRAcc}(C1 \leftarrow Cond) = \frac{n'(Cond)}{N'} \left( \frac{n'(Cl,Cond)}{n'(Cond)} - \frac{n'(Cl)}{N'} \right)
  \]
  
  - \( N' \): sum of weights of examples
  - \( n'(Cond) \): sum of weights of all covered examples
  - \( n'(Cl,Cond) \): sum of weights of all correctly covered examples

Part IV. Descriptive DM techniques

- Predictive vs. descriptive induction
- Subgroup discovery
- Association rule learning
- Hierarchical clustering

Association Rule Learning

Rules: \( X \Rightarrow Y \), if \( X \) then \( Y \)

\( X \) and \( Y \) are itemsets (records, conjunction of items), where items/features are binary-valued attributes

Given: Transactions

\begin{array}{ccc}
\text{Itemsets (records)} & i_1 & i_2 \\
\hline
1 & 1 & 0 \\
1 & 0 & 1 \\
\hline
\end{array}

Find: A set of association rules in the form \( X \Rightarrow Y \)

Example: Market basket analysis

- beer & coke \( \Rightarrow \) peanuts & chips (0.05, 0.65)
  
  - Support: \( \text{Sup}(X,Y) = \#XY/\#D = p(XY) \)
  - Confidence: \( \text{Conf}(X,Y) = \#XY/\#X = \text{Sup}(X,Y)/\text{Sup}(X) = p(XY)/p(X) = p(Y|X) \)

Association Rule Learning: Examples

- Market basket analysis
  
  - beer & coke \( \Rightarrow \) peanuts & chips (5%, 65%)
  
  - Support 5%: 5% of all customers buy all four items
  
  - Confidence 65%: 65% of customers that buy beer and coke also buy peanuts and chips

- Insurance
  
  - mortgage & loans & savings \( \Rightarrow \) insurance (2%, 62%)
  
  - Support 2%: 2% of all customers have all four
  
  - Confidence 62%: 62% of all customers that have mortgage, loan and savings also have insurance
**Association rule learning**

- **X ⇒ Y . . . IF X THEN Y**, where X and Y are itemsets
- intuitive meaning: transactions that contain X tend to contain Y
- **Items** - binary attributes (features) m,f, headache, muscle pain, arthrotic, arthritic, spondylotic, spondylitic, stiff_less_1_hour
- **Example transactions** - itemsets formed of patient records

```
i1    i2  ……  … i50
1    0    0  …  0
0    1    0  …  0
…    …    …  …  …
```

- **Association rules**
  - spondylotic ⇒ arthritic & stiff_gt_1_hour [5%, 70%]
  - arthrotic & spondylotic ⇒ stiff_less_1_hour [20%, 90%]

---

**Association Rule Learning**

**Given:** a set of transactions D

**Find:** all association rules that hold on the set of transactions that have

- user defined minimum support, i.e., support > MinSup, and
- user defined minimum confidence, i.e., confidence > MinConf

It is a form of exploratory data analysis, rather than hypothesis verification

---

**Searching for the associations**

- Find all large itemsets
- Use the large itemsets to generate association rules
- If XY is a large itemset, compute
  \[ r = \frac{\text{support}(XY)}{\text{support}(X)} \]
- If \( r > \text{MinConf} \), then \( X \Rightarrow Y \) holds (support > MinSup, as XY is large)

---

**Large itemsets**

- Large itemsets are itemsets that appear in at least MinSup transaction
- All subsets of a large itemset are large itemsets (e.g., if A,B appears in at least MinSup transactions, so do A and B)
- This observation is the basis for very efficient algorithms for association rules discovery (linear in the number of transactions)

---

**Association vs. Classification rules**

- Exploration of dependencies
- Different combinations of dependent and independent attributes
- Complete search (all rules found)
- Focused prediction
  - Predict one attribute (class) from the others
  - Heuristic search (subset of rules found)

---

**Part IV. Descriptive DM techniques**

- Predictive vs. descriptive induction
- Subgroup discovery
- Association rule learning
- Hierarchical clustering
Hierarchical clustering

• Algorithm (agglomerative hierarchical clustering):

Each instance is a cluster;
repeat
  find nearest pair Ci in Cj;
  fuse Ci in Cj in a new cluster Cr = Ci U Cj;
  determine dissimilarities between Cr and other clusters;
until one cluster left;

• Dendogram:

Hierarchical clustering

• Fusing the nearest pair of clusters

• Minimizing intra-cluster similarity
• Maximizing inter-cluster similarity
• Computing the dissimilarities from the “new” cluster

Hierarchical clustering: example

Results of clustering

A dendogram of resistance vectors


Part V: Relational Data Mining

Learning as search
• What is RDM?
• Propositionalization techniques
• Inductive Logic Programming

Learning as search

• Structuring the state space: Representing a partial order of hypotheses (e.g., rules) as a graph
  – nodes: concept descriptions (hypotheses/rules)
  – arcs defined by specialization/generalization operators: an arc from parent to child exists if-and-only-if parent is a proper most specific generalization of child

• Specialization operators: e.g., adding conditions: s(A=a2 & B=b1) = {A=a2 & B=b1 & D=d1, A=a2 & B=b1 & D=d2}
• Generalization operators: e.g., dropping conditions: g(A=a2 & B=b1) = {A=a2, B=b1}
• Partial order of hypotheses defines a lattice (called a refinement graph)
Learn-one-rule as search - Structuring the hypothesis space: PlayTennis example

Play tennis = yes
IF Wind=weak
Play tennis = yes
IF Wind=strong
Play tennis = yes
IF Humidity=normal
Play tennis = yes
IF Humidity=high
Play tennis = yes
IF Humidity=normal,
   Wind=weak
Play tennis = yes
IF Humidity=normal,
   Wind=strong
Play tennis = yes
IF Humidity=normal,
   Outlook=sunny
Play tennis = yes
IF Humidity=normal,
   Outlook=rain
...

Learn-one-rule as heuristic search: PlayTennis example

Play tennis = yes
[9+,5−] (14)
[6+,2−] (8)
[3+,3−] (6) [6+,1−] (7)
[3+,4−] (7)
[2+,0−] (2)

Learning as search (Mitchell’s version space model)

- Hypothesis language LH defines the state space
- How to structure the hypothesis space LH?
- How to move from one hypothesis to another?

The version space: region between S (maximally specific) and G (maximally general) complete and consistent concept descriptions

too general
more general
complete and consistent

Learning as search: Learner’s ingredients

- structure of the search space (specialization and generalization operators)
- search strategy
  - depth-first
  - breadth-first
  - heuristic search (best first, hill-climbing, beam search)
- search heuristics
  - measure of attribute ‘informativity’
  - measure of ‘expected classification accuracy’ (relative frequency, Laplace estimate, m-estimate), ...
  - stopping criteria (consistency, completeness, statistical significance, …)

Learn-one-rule: search heuristics

- Assume a two-class problem
- Two classes (+,-), learn rules for + class (Cl).
- Search for specializations R’ of a rule R = Cl ← Cond from the RuleBase.
- Specialization R’ of rule R = Cl ← Cond has the form R’ = Cl ← Cond & Cond
- Heuristic search for rules: find the ‘best’ Cond to be added to the current rule R, such that rule accuracy is improved, e.g., such that Acc(R’) > Acc(R)
- where the expected classification accuracy can be estimated as A(R) = p(Cl|Cond)
Learn-one-rule – Search strategy: Greedy vs. beam search

- learn-one-rule by greedy general-to-specific search, at each step selecting the ‘best’ descendant, no backtracking
  - e.g., the best descendant of the initial rule
    - `PlayTennis = yes ← Humidity=normal`
- beam search: maintain a list of k best candidates at each step; descendants (specializations) of each of these k candidates are generated, and the resulting set is again reduced to k best candidates

Part V: Relational Data Mining

- Learning as search
  - What is RDM?
- Propositionalization techniques
- Inductive Logic Programming

Predictive relational DM

- Data stored in relational databases
- Single relation - propositional DM
  - example is a tuple of values of a fixed number of attributes (one attribute is a class)
  - example set is a table (simple field values)
- Multiple relations - relational DM (ILP)
  - example is a tuple or a set of tuples (logical fact or set of logical facts)
  - example set is a set of tables (simple or complex structured objects as field values)

Data for propositional DM

Sample single relation data table

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>City</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
</tr>
</thead>
<tbody>
<tr>
<td>3478</td>
<td>Boston</td>
<td>10000</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
<td>0.5</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
</tr>
<tr>
<td>3479</td>
<td>Chicago</td>
<td>20000</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
<td>0.6</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
</tr>
</tbody>
</table>

Customer table including order and store information.

Multi-relational data made propositional

- Sample relation table

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>City</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
</tr>
</thead>
<tbody>
<tr>
<td>3478</td>
<td>Boston</td>
<td>10000</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
<td>0.5</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
</tr>
<tr>
<td>3479</td>
<td>Chicago</td>
<td>20000</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
<td>0.6</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>City</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
</tr>
</thead>
<tbody>
<tr>
<td>3478</td>
<td>Boston</td>
<td>10000</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
<td>0.5</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
</tr>
<tr>
<td>3479</td>
<td>Chicago</td>
<td>20000</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
<td>0.6</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CustomerID</th>
<th>City</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
<th>balance</th>
<th>婚姻</th>
<th>income</th>
<th>annualExp</th>
<th>employed</th>
<th>education</th>
</tr>
</thead>
<tbody>
<tr>
<td>3478</td>
<td>Boston</td>
<td>10000</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
<td>0.5</td>
<td>Single</td>
<td>50000</td>
<td>20000</td>
<td>Yes</td>
<td>Master</td>
</tr>
<tr>
<td>3479</td>
<td>Chicago</td>
<td>20000</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
<td>0.6</td>
<td>Married</td>
<td>70000</td>
<td>40000</td>
<td>No</td>
<td>Bachelor</td>
</tr>
</tbody>
</table>

Customer table with multiple entries.

Relational Data Mining (ILP)

- Learning from multiple tables
- Complex relational problems:
  - temporal data: time series in medicine, traffic control, ...
  - structured data: representation of molecules and their properties in protein engineering, biochemistry, ...
Basic Relational Data Mining tasks

Predictive RDM

- Positive examples \( E^+ \)
- Negative examples \( E^- \)
- Background knowledge \( B \)
- Hypothesis language \( L_H \)
- Covers relation

Descriptive RDM

- Positive examples \( E^+ \)
- Negative examples \( E^- \)
- Background knowledge \( B \)
- Hypothesis language \( L_H \)
- Covers relation

Sample problem

Knowledge discovery

\[ E^+ = \{ \text{daughter(mary,ann)}, \text{daughter(eve,tom)} \} \]
\[ E^- = \{ \text{daughter(tom,ann)}, \text{daughter(eve,ann)} \} \]
\[ B = \{ \text{mother(ann,mary)}, \text{mother(ann,tom)}, \text{father(tom,eve)}, \text{father(tom,ian)}, \text{female(ann)}, \text{female(mary), female(eve)}, \text{male(pat)}, \text{male(tom)}, \text{parent(X,Y) \leftarrow mother(X,Y)}, \text{parent(X,Y) \leftarrow father(X,Y)} \} \]

Sample problem

Knowledge discovery

Predictive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Negative examples \( E^- \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive and no negative examples

Descriptive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive examples

Sample problem

Knowledge discovery

\[ E^+ = \{ \text{daughter(mary,ann)}, \text{daughter(eve,tom)} \} \]
\[ E^- = \{ \text{daughter(tom,ann)}, \text{daughter(eve,ann)} \} \]
\[ B = \{ \text{mother(ann,mary)}, \text{mother(ann,tom)}, \text{father(tom,eve)}, \text{father(tom,ian)}, \text{female(ann)}, \text{female(mary), female(eve)}, \text{male(pat)}, \text{male(tom)}, \text{parent(X,Y) \leftarrow mother(X,Y)}, \text{parent(X,Y) \leftarrow father(X,Y)} \} \]

Sample problem

Knowledge discovery

Predictive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Negative examples \( E^- \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive and no negative examples

Descriptive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive examples

Sample problem

Knowledge discovery

\[ E^+ = \{ \text{daughter(mary,ann)}, \text{daughter(eve,tom)} \} \]
\[ E^- = \{ \text{daughter(tom,ann)}, \text{daughter(eve,ann)} \} \]
\[ B = \{ \text{mother(ann,mary)}, \text{mother(ann,tom)}, \text{father(tom,eve)}, \text{father(tom,ian)}, \text{female(ann)}, \text{female(mary), female(eve)}, \text{male(pat)}, \text{male(tom)}, \text{parent(X,Y) \leftarrow mother(X,Y)}, \text{parent(X,Y) \leftarrow father(X,Y)} \} \]

Sample problem

Knowledge discovery

Predictive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Negative examples \( E^- \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive and no negative examples

Descriptive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive examples

Sample problem

Knowledge discovery

\[ E^+ = \{ \text{daughter(mary,ann)}, \text{daughter(eve,tom)} \} \]
\[ E^- = \{ \text{daughter(tom,ann)}, \text{daughter(eve,ann)} \} \]
\[ B = \{ \text{mother(ann,mary)}, \text{mother(ann,tom)}, \text{father(tom,eve)}, \text{father(tom,ian)}, \text{female(ann)}, \text{female(mary), female(eve)}, \text{male(pat)}, \text{male(tom)}, \text{parent(X,Y) \leftarrow mother(X,Y)}, \text{parent(X,Y) \leftarrow father(X,Y)} \} \]

Sample problem

Knowledge discovery

Predictive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Negative examples \( E^- \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive and no negative examples

Descriptive ILP

- Given:
  - A set of observations
  - Positive examples \( E^+ \)
  - Background knowledge \( B \)
  - Hypothesis language \( L_H \)
  - Covers relation

- Find:
  - A hypothesis \( H \in L_H \) such that (given \( B \)) \( H \) covers all positive examples

Sample problem

Knowledge discovery

\[ E^+ = \{ \text{daughter(mary,ann)}, \text{daughter(eve,tom)} \} \]
\[ E^- = \{ \text{daughter(tom,ann)}, \text{daughter(eve,ann)} \} \]
\[ B = \{ \text{mother(ann,mary)}, \text{mother(ann,tom)}, \text{father(tom,eve)}, \text{father(tom,ian)}, \text{female(ann)}, \text{female(mary), female(eve)}, \text{male(pat)}, \text{male(tom)}, \text{parent(X,Y) \leftarrow mother(X,Y)}, \text{parent(X,Y) \leftarrow father(X,Y)} \} \]
Sample problem
Logic programming

\[ E^+ = \{ \text{sort([2,1,3],[1,2,3])} \} \]
\[ E^- = \{ \text{sort([2,1],[1]),sort([3,1,2],[2,1,3])} \} \]

\[ B : \text{definitions of permutation/2 and sorted/1} \]

- Predictive ILP
  \[ \text{sort}(X,Y) \leftarrow \text{permutation}(X,Y), \text{sorted}(Y). \]

- Descriptive ILP
  \[ \text{sorted}(Y) \leftarrow \text{sort}(X,Y). \]
  \[ \text{permutation}(X,Y) \leftarrow \text{sort}(X,Y) \]
  \[ \text{sorted}(X) \leftarrow \text{sort}(X,X) \]

RDM knowledge representation (database)

ER diagram for East-West trains

ILP representation: Datalog ground facts

- Example: `eastbound(t1).`
- Background theory:
  \[ \text{car}(t1,c1). \]
  \[ \text{rectangle}(c1). \]
  \[ \text{short}(c1). \]
  \[ \text{none}(c1). \]
  \[ \text{two_wheels}(c1). \]
  \[ \text{load}(c1,l1). \]
  \[ \text{circle}(l1). \]
  \[ \text{one_load}(l1). \]
- Hypothesis (predictive ILP):
  \[ \text{eastbound}(T) :- \text{car}(T,C), \text{short}(C), \neg \text{none}(C). \]

ILP representation: Datalog ground clauses

- Example:
  \[ \text{eastbound}(t1):- \text{car}(t1,c1), \text{rectangle}(c1), \text{short}(c1), \neg \text{none}(c1), \text{two_wheels}(c1), \text{load}(c1,l1), \text{circle}(l1), \text{one_load}(l1). \]
  \[ \text{car}(t1,c2). \]
  \[ \text{rectangle}(c2). \]
  \[ \text{long}(c2). \]
  \[ \text{none}(c2). \]
  \[ \text{three_wheels}(c2). \]
  \[ \text{load}(c2,l2), \text{hexagon}(l2), \text{one_load}(l2). \]
  \[ \text{car}(t1,c3). \]
  \[ \text{rectangle}(c3). \]
  \[ \text{short}(c3). \]
  \[ \text{peaked}(c3). \]
  \[ \text{two_wheels}(c3). \]
  \[ \text{load}(c3,l3), \text{triangle}(l3), \text{one_load}(l3). \]
  \[ \text{car}(t1,c4). \]
  \[ \text{rectangle}(c4). \]
  \[ \text{long}(c4). \]
  \[ \text{none}(c4). \]
  \[ \text{two_wheels}(c4). \]
  \[ \text{load}(c4,l4), \text{rectangle}(l4), \text{three_loads}(l4). \]
- Background theory: empty
- Hypothesis:
  \[ \text{eastbound}(T) :- \neg \text{car}(T,C), \neg \text{short}(C), \neg \text{none}(C). \]
ILP representation: Prolog terms

• Example:
eastbound([c(rectangle,short,none,2,l(circle,1)),
c(rectangle,long,none,3,l(hexagon,1)),
c(rectangle,short,peaked,2,l(triangle,1)),
c(rectangle,long,none,2,l(rectangle,3))]).

• Background theory: member/2, arg/3

• Hypothesis:
eastbound(T):-member(C,T),arg(2,C,short), not arg(3,C,none).

First-order representations

• Propositional representations:
  – database is fixed-size vector of values
  – features are those given in the dataset

• First-order representations:
  – database is flexible-size, structured object
    • sequence, set, graph
    • hierarchical: e.g. set of sequences
  – features need to be selected from potentially infinite set

Complexity of RDM problems

• Simplest case: single table with primary key
  – example corresponds to tuple of constants
  – attribute-value or propositional learning

• Next: single table without primary key
  – example corresponds to set of tuples of constants
  – multiple-instance problem

• Complexity resides in many-to-one foreign keys
  – lists, sets, multisets
  – non-determinate variables

Part V:
Relational Data Mining

• Learning as search
• What is RDM?
  Propositionalization techniques
  • Inductive Logic Programming

Rule learning: The standard view

• Hypothesis construction: find a set of \( n \) rules
  – usually simplified by \( n \) separate rule constructions
    • exception: HYPER

• Rule construction: find a pair (Head, Body)
  – e.g. select head (class) and construct body by searching the VersionSpace
    • exceptions: CN2, APRIORI

• Body construction: find a set of \( m \) literals
  – usually simplified by adding one literal at a time
    • problem (ILP): literals introducing new variables

Rule learning revisited

• Hypothesis construction: find a set of \( n \) rules

• Rule construction: find a pair (Head, Body)

• Body construction: find a set of \( m \) features
  – Features can be either defined by background knowledge or constructed through constructive induction
  – In propositional learning features may increase expressiveness through negation
  – Every ILP system does constructive induction

• Feature construction: find a set of \( k \) literals
  – finding interesting features is discovery task rather than classification
task e.g. interesting subgroups, frequent itemsets
  – excellent results achieved also by feature construction through predictive propositional learning and ILP (Srinivasan)
### First-order feature construction

- All the expressiveness of ILP is in the features
- Given a way to construct (or choose) first-order features, body construction in ILP becomes propositional
  - idea: learn non-determinate clauses with LINUS by saturating background knowledge (performing systematic feature construction in a given language bias)

### Representation issues (1)

- In the database and Datalog ground fact representations individual examples are not easily separable
- Term and Datalog ground clause representations enable the separation of individuals
- Term representation collects all information about an individual in one structured term

### Declarative bias for first-order feature construction

- In ILP, features involve interactions of local variables
- Features should define properties of individuals (e.g. trains, molecules) or their parts (e.g., cars, atoms)
- Feature construction in LINUS, using the following language bias:
  - one free global variable (denoting an individual, e.g. train)
  - one or more structural predicates: (e.g., has_car(T,C)), each introducing a new existential local variable (e.g. car, atom), using either the global variable (train, molecule) or a local variable introduced by other structural predicates (car, load)
  - one or more utility predicates defining properties of individuals or their parts: no new variables, just using variables
  - all variables should be used
  - parameter: max. number of predicates forming a feature

### Representation issues (2)

- Term representation provides strong language bias
- Term representation can be flattened to be described by ground facts, using
  - structural predicates (e.g. car(t1,c1), load(c1,l1)) to introduce substructures
  - utility predicates, to define properties of individuals (e.g. long(t1)) or their parts (e.g., long(c1), circle(l1)).
- This observation can be used as a language bias to construct new features

### Sample first-order features

- The following rule has two features ‘has a short car’ and ‘has a closed car’:
  
  ```
  eastbound(T):-hasCar(T,C1),clength(C1,short),
  hasCar(T,C2),not croof(C2,none).
  ```

- The following rule has one feature ‘has a short closed car’:
  
  ```
  eastbound(T):-hasCar(T,C),clength(C,short),
  not croof(C,none).
  ```

- Equivalent representation:
  
  ```
  eastbound(T):-hasShortCar(T),hasClosedCar(T).
  ```
  
  ```
  hasShortCar(T):-hasCar(T,C),clength(C,short).
  ```
  
  ```
  hasClosedCar(T):-hasCar(T,C),not croof(C,none).
  ```
Propositionalization in a nutshell

Propositionalization task
- Transform a multi-relational (multiple-table) representation to a propositional representation (single table)

Proposed in ILP systems LINUS (1991), IBP (1999), ...

Linus revisited

- Standard LINUS:
  - transforming an ILP problem to a propositional problem
  - apply background knowledge predicates
- Revisited LINUS:
  - Systematic first-order feature construction in a given language bias
  - Too many features?
    - use a relevancy filter (Gamberger and Lavrac)

Linus revisited: Example: East-West trains

Rules induced by CN2, using 190 first-order features with up to two utility predicates:

- eastbound(T):- westbound(T):-
- hasCarHasLoadSingleTriangle(T), not hasCarEllipse(T),
- not hasCarLongJagged(T), not hasCarShortFlat(T),
- hasCarHasLoadCircle(T), not hasCarPeakedTwo(T).

Meaning:
- eastbound(T):-
- hasCar(T,C), hasLoad(C,L), lshape(L,tri), lnumber(L,1),
- not hasCar(T,C3), hasLoad(C3,L3), lshape(L3,circ).
- westbound(T):-
- not hasCar(T,C1), lshape(L1,tri),
- not hasCar(T,C2), lshape(L2,circ),
- not hasCar(T,C3), croof(C3,peak), cwheels(C3,2)).

Part V: Relational Data Mining

- Learning as search
- What is RDM?
- Propositionalization techniques

Inductive Logic Programming

ILP as search of program clauses

- An ILP learner can be described by
  - the structure of the space of clauses
    - based on the generality relation
    - Let C and D be two clauses.
    - C is more general than D (C ⊃ D) iff covers(D) ⊂ covers(C)
    - Example: p(X,Y) ← n(Y,X) is more general than p(X,Y) ← n(Y,X), q(X)
  - its search strategy
    - uninformed search (depth-first, breadth-first, iterative deepening)
    - heuristic search (best-first, hill-climbing, beam search)
    - its heuristics
      - for directing search
      - for stopping search (quality criterion)
ILP as search of program clauses

• **Semantic generality**
  Hypothesis \( H_1 \) is semantically more general than \( H_2 \) w.r.t. background theory \( B \) if and only if \( B \cup H_1 \models H_2 \)

• **Syntactic generality or \( \theta \)-subsumption**
  (most popular in ILP)
  – Clause \( c_1 \) \( \theta \)-subsumes \( c_2 \) (\( c_1 \geq \theta c_2 \))
    if and only if \( \exists \theta : c_1 \theta \subseteq c_2 \)
  – Hypothesis \( H_1 \geq \theta H_2 \)
    if and only if \( \forall c_2 \in H_2 \) exists \( c_1 \in H_1 \) such that \( c_1 \geq \theta c_2 \)

• **Example**
  \( c_1 = \text{daughter}(X,Y) \leftarrow \text{parent}(Y,X) \)
  \( c_2 = \text{daughter}(\text{mary},\text{ann}) \leftarrow \text{female}(\text{mary}), \text{parent}(\text{ann},\text{mary}), \text{parent}(\text{ann},\text{tom}) \)
  \( c_1 \theta \)-subsumes \( c_2 \) under \( \theta = \{X/\text{mary}, Y/\text{ann}\} \)

The role of subsumption in ILP

• Generality ordering for hypotheses
• Pruning of the search space:
  – generalization
    • if C covers a neg. example then its generalizations need not be considered
  – specialization
    • if C doesn’t cover a pos. example then its specializations need not be considered
• Top-down search of refinement graphs
• Bottom-up search of the hypo. space by
  – building least general generalizations, and
  – inverting resolutions

Structuring the hypothesis space

Two strategies for learning

• General-to-specific
  – if \( \theta \)-subsumption is used then refinement operators
• Specific-to-general search
  – if \( \theta \)-subsumption is used then lgg-operator or generalization operator

ILP as search of program clauses

• Two strategies for learning
  – Top-down search of refinement graphs
  – Bottom-up search
    • building least general generalizations
    • inverting resolution (CIGOL)
    • inverting entailment (PROGOL)
Generality ordering of clauses

<table>
<thead>
<tr>
<th>Training examples</th>
<th>Background knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>daughter(mary,ann).</td>
<td>parent(ann,mary). female(ann).</td>
</tr>
<tr>
<td>daughter(eve,tom).</td>
<td>parent(ann,mary). female(mary).</td>
</tr>
<tr>
<td>daughter(tom,ann).</td>
<td>parent(tom,eve). female(eve).</td>
</tr>
<tr>
<td>daughter(eve,ann).</td>
<td>parent(tom,ann).</td>
</tr>
</tbody>
</table>


Greedy search of the best clause

<table>
<thead>
<tr>
<th>Training examples</th>
<th>Background knowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>daughter(mary,ann).</td>
<td>parent(ann,mary). female(ann).</td>
</tr>
<tr>
<td>daughter(eve,ann).</td>
<td>parent(ann,mary). female(mary).</td>
</tr>
<tr>
<td>daughter(tom,ann).</td>
<td>parent(tom,eve). female(eve).</td>
</tr>
<tr>
<td>daughter(eve,ann).</td>
<td>parent(tom,ann).</td>
</tr>
</tbody>
</table>

Part of the refinement graph for the family relations problem.

FOIL

- Language: function-free normal programs
  - recursion, negation, new variables in the body, no functors, no constants (original)
- Algorithm: covering
- Search heuristics: weighted info gain
- Search strategy: hill climbing
- Stopping criterion: encoding length restriction
- Search space reduction: types, in/out modes
determinate literals
- Ground background knowledge, extensional coverage
- Implemented in C

Part V: Summary

- RDM extends DM by allowing multiple tables describing structured data
- Complexity of representation and therefore of learning is determined by one-to-many links
- Many RDM problems are individual-centred and therefore allow strong declarative bias