Logical Decision Trees for Classification, Regression and Clustering

Hendrik Blockeel
Katholieke Universiteit Leuven
Introduction

- Many RDM techniques based on rule induction (e.g. classical ILP)
- Another popular symbolic induction approach: induction of decision trees
- Can we use this approach for RDM?
  - Starting point: well-known techniques for induction of decision trees from data
  - Extend them in two ways
    - View of “predictive clustering” trees
      - Generalises over classification and regression trees
    - Make them work on relational data
      - Introducing First Order Logical Decision Trees
Overview

Predictive clustering (generalise over several induction tasks)

First order logical decision trees (make trees usable in ILP)

Top-down induction of first order logical decision trees
1. Predictive Clustering

- Predictive clustering
  (generalise over several induction tasks)

- First order logical decision trees
  (make trees usable in ILP)

- Top-down induction of first order logical decision trees
Typical Induction Tasks

- classification
  - predicting symbolic values
- regression
  - predicting numbers
- clustering
  - finding structure in data (clusters of similar objects)
- ...
Clustering

All data

clus1
clus2
clus3

1a
1b
Predictive Clustering

- Find clusters in data
- Partition instance spaces into regions corresponding to clusters
- Different predictive model for each region
  - typically: prototype
Using Clusters for Prediction

- 2-step prediction process
  - assign new instance to cluster
  - make prediction based on cluster information
Examples

- Rule based systems
  - body of rule describes cluster of examples
  - head contains prediction

- Tree based systems
  - tests form description of leaves (clusters)
  - leaves contain prediction

- Other uses possible as well
  - e.g. distance-based assignment
  - predict description
Creating Clusters for Prediction

- Form clusters with a specific kind of prediction in mind
- Observation:
  - classification
  - regression
  - flexible prediction
  - unsupervised learning

are special cases
E.g.: Finding a classification rule

=> +
E.g.: Finding a classification rule

blue => +
E.g.: Finding a classification rule

blue, large => +
E.g.: Finding a classification rule
E.g.: Finding a classification rule
E.g.: Finding a classification rule

blue, large => +
Quality Criteria for Predictive Clustering

- minimise variance within clusters
- variance based on distance measure
- setting depends on distance:
  - *distance in* $P$: classification / regression
  - *distance in* $D$: unsupervised learning
  - *distance in* $I = P \times D$: flexible prediction, multiple prediction, ...
How General Is Predictive Clustering?

- Both rule-based and tree-based methods
- Model associated with cluster can be
  - constant
  - linear model
  - ...
- Assignment of instance to cluster can be
  - based on *intensional description* of cluster
  - based on *distance* to (examples in) cluster
- Encompasses several modified inductive systems
  - Semi-supervised systems: Emde
  - Hybrid approaches: Domingos, Webb, ...
Using Decision Trees for Predictive Clustering

- Decision tree = *mapping* example → result
- Also naturally represents *cluster hierarchy*
Induction of Decision Trees

(Quinlan, 1986)

**function** TDIDT(E: set of examples) **returns** decision tree:

1. $T :=$ set of possible tests;
2. $\tau :=$ BEST_SPLIT($T$, $E$);
3. $\mathcal{E} :=$ partition induced on $E$ by $\tau$
4. **if** STOP_CRIT($\mathcal{E}$, $E$) **then**
   - **return** leaf(INFO($E$))
5. **else**
   - **for all** $E_i$ in $\mathcal{E}$: $t_i :=$ TDIDT($E_i$)
   - **return** inode($\tau$, $\{(i, t_i)\}$)
## Induction of Decision Trees

<table>
<thead>
<tr>
<th>Type</th>
<th>BEST-SPLIT</th>
<th>STOP-CRIT</th>
<th>INFO</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>classification</strong></td>
<td>Gain(ratio); Gini; ...</td>
<td>$\chi^2$-test; MDL</td>
<td>mode</td>
</tr>
<tr>
<td><strong>regression</strong></td>
<td>Variance in P</td>
<td>F-test; MDL</td>
<td>mean</td>
</tr>
<tr>
<td><strong>clustering</strong></td>
<td>Variance in I or D</td>
<td>F-test; MDL</td>
<td>prototype; set of examples</td>
</tr>
</tbody>
</table>
Conclusions (predictive clustering)

- Predictive clustering is a relatively general task, encompassing several more specific tasks
- Induction of decision trees is a suitable technique for predictive clustering
Predictive clustering (generalise over several induction tasks)

First order logical decision trees (make trees usable in ILP)

Top-down induction of first order logical decision trees
Attribute Value Framework

- Example description = values for fixed vector of attributes
- Relatively unstructured
  - properties of the whole, not of arbitrary components

<table>
<thead>
<tr>
<th>Shape</th>
<th>Color</th>
<th>Size</th>
<th>Weight</th>
<th>Taste</th>
</tr>
</thead>
<tbody>
<tr>
<td>round</td>
<td>green</td>
<td>250 cm³</td>
<td>150g</td>
<td>sour</td>
</tr>
</tbody>
</table>
Inductive Logic Programming

- Structured example descriptions
- Theories represented in first order predicate logic instead of propositional logic

\{\text{object}(o1), \text{object}(o2), \text{object}(o3), \text{square}(o1), \text{triangle}(o2), \text{circle}(o3), \text{inside}(o2,o3), \text{blue}\}

\{\text{object}(o1), \text{object}(o2), \text{object}(o3), \text{triangle}(o1), \text{square}(o2), \text{circle}(o3), \text{inside}(o2,o3), \text{red}\}

\text{triangle}(x), \text{circle}(y), \text{inside}(x,y) \Rightarrow \text{blue}
Relational Data Mining

<table>
<thead>
<tr>
<th>Name</th>
<th>Job</th>
<th>Speed</th>
<th>Fine?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>teacher</td>
<td>150</td>
<td>Y</td>
</tr>
<tr>
<td>Bob</td>
<td>politician</td>
<td>160</td>
<td>N</td>
</tr>
<tr>
<td>Chris</td>
<td>engineer</td>
<td>120</td>
<td>N</td>
</tr>
<tr>
<td>Dave</td>
<td>writer</td>
<td>155</td>
<td>N</td>
</tr>
<tr>
<td>Earnest</td>
<td>politician</td>
<td>120</td>
<td>N</td>
</tr>
</tbody>
</table>

**KNOWS**

<table>
<thead>
<tr>
<th>Name1</th>
<th>Name2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ann</td>
<td>Chris</td>
</tr>
<tr>
<td>Ann</td>
<td>Dave</td>
</tr>
<tr>
<td>Bob</td>
<td>Earnest</td>
</tr>
<tr>
<td>Chris</td>
<td>Dave</td>
</tr>
<tr>
<td>Dave</td>
<td>Bob</td>
</tr>
</tbody>
</table>

(AVL) Speed > 120 and Job ≠ politician => Fine? = Y

(ILP) speed(x,s), s > 120, not job(x, politician), not ∃y:(knows(x,y), job(y,politician)) => fine(x,Y)
Earlier work on structural decision trees by Watanabe (1991), Kramer (1996), ...

Here focus on “first order logical decision trees”

- Binary trees
- Each node contains a conjunction of literals
  - Free variables existentially quantified
- Nodes can share variables
- Test in node = conjunction in node + conjunctions on path from root to node
- Complex semantics
- High expressiveness
Example FOLDT

\[
(\forall x: \neg \text{worn}(x)) \implies \text{ok}
\]

\[
(\exists x: \text{worn}(x) \land \text{irreplaceable}(x)) \implies \text{sendback}
\]

\[
(\exists x \forall y: \text{worn}(x) \land \neg (\text{worn}(y) \land \text{irreplaceable}(y))) \implies \text{fix}
\]
Expressiveness

FOL formula equivalent with tree:

(∀x: ¬ worn(x)) => ok
(∃x: worn(x) ∧ irreplaceable(x)) => sendback
(∃x∀y: worn(x) ∧ ¬(worn(y) ∧ irreplaceable(y))) => fix

Logic program equivalent with tree:

a ← worn(X)
b ← worn(X), irreplaceable(X)
ok ← ¬ a
sendback ← b
fix ← a ∧ ¬ b
speed(x,s), s > 120, not job(x, politician), not (knows(x,y), job(y,politician)) => fine(x,Y)
Expressiveness

- Difference is specific for first-order case
- Possible remedies for ILP systems:
  - invent auxiliary predicates
  - use both $\forall$ and $\exists$
  - induce “decision lists”

\[ F = \text{Flat logic programs} \]
\[ T = \text{decision Trees} \]
\[ L = \text{decision Lists} \]
Conclusions (FOLDTs)

- FOLDTs have high expressiveness
  - Ability to cope with exceptions
  - Similar to decision lists
  - Related to predicate invention + negation, mixed quantification of variables
- Relatively straightforward mapping RDB -> FOL
3: Induction of FOLDTs

- Predictive clustering (generalise over several induction tasks)
- First order logical decision trees (make trees usable in ILP)

Top-down induction of first order logical decision trees
The “Upgrading” Methodology for Building ILP Systems

- Observation: AVL contains many advanced, specialised techniques
- Build ILP system from existing AVL system
  - Take an AVL system
  - Identify necessary changes for ILP
  - Implement changes
- See De Raedt & Van Laer, RDM Ch. 10
Top-down Induction of FOLDTs

- One implementation: TILDE system
  - Originally based on C4.5 (Quinlan 1993)
  - Adaptations to ILP context
    - Refinement operator under $\theta$-subsumption
  - Instantiations of TDIDT for classification, regression and clustering
  - Several variants implemented
    - E.g. level-wise tree building (Mehta et al. 1996)

- Experimental evaluation on many tasks
  - Usually high accuracy & efficiency
Foldts in RDB Context

- Most ILP systems have relatively complicated bias specifications, Tilde is no different.
- One possible way to go:
  - Move away from logic, towards more generally known languages (UML, relational databases, ...)
  - See Knobbe et al., PKDD-2000 for an example approach
    - Language bias based on UML description of database
    - Graphical representation of patterns

\[
\text{Person} \xrightarrow{\text{Speed}>120, \text{Job} <> \text{Politician}} \text{Knows} \xrightarrow{\text{Job} = \text{politician}} \text{Person}
\]
Some Uses of FOLDTs

- Standard classification / regression tasks
- Multiple prediction
  - River water quality
- Relational ranking
  - E.g. In meta-learning setting: predict which learner will perform best, 2nd, 3rd, ... on some task
- Hierarchical multi-classification
  - Predict set of classes, where classes ordered in hierarchy
- ...
- ...
Multiple Prediction

\[
\begin{align*}
\text{abundance}(\text{Tubifex sp.}, 5) &? \\
\text{yes} &\quad &\text{abundance}(\text{Sphaerotilus natans}, 5) &? \\
\text{no} &\quad &\text{abundance}(\ldots) &?
\end{align*}
\]

\[
\begin{array}{l}
T = 0.357111 \\
pH = -0.496808 \\
\text{cond} = 1.23151 \\
O2 = -1.09279 \\
O2sat = -1.04837 \\
CO2 = 0.893152 \\
hard = 0.988909 \\
NO2 = 0.54731 \\
NO3 = 0.426773 \\
NH4 = 1.11263 \\
PO4 = 0.875459 \\
Cl = 0.997237 \\
SiO2 = 0.97025 \\
KMnO4 = 1.29711 \\
K2Cr2O7 = 0.97025 \\
BOD = 0.67012
\end{array}
\]

<- "standardized" values (how many standard deviations above mean)
Predicting Saprobiic Index

- Allow tests for max, min, avg over period
- Let TILDE decide on optimal period length for each individual test

```
saprobe_index(A,B,C,D,E)
get_avg(o2sat,70,A,D,C,B,F) , F < 64.6 ?
  +--yes: [2.7485]
  +--no: get_min(o2sat,190,A,D,C,B,G) , G < 64.6 ?
    +--yes: [2.536]
    +--no: get_avg(k2cr2o7,70,A,D,C,B,H) , H < 22.9 ?
      +--yes: get_avg(conduct,190,A,D,C,B,I) , I < 191 ?
        +--yes: [1.56913]
        +--no: get_avg(bod,100,A,D,C,B,J) , J < 5.9 ?
          +--yes: get_max(k2cr2o7,70,A,D,C,B,K) , K < 3.9 ?
            +--yes: get_max(temp,190,A,D,C,B,L) , L < 11.9 ?
              +--yes: get_max(sio2,190,A,D,C,B,M) , M < 1.9 ?
                +--yes: [1.54182]
                +--no: [1.6665]
            +--no: [1.75091]
        +--no: [1.75091]
    +--no: [1.75091]
```

Relational Ranking

er_ranks(A,B,C,D,E,F,G,H,I,J,K)
classvalue_freq(A,L,M),M<0.165 ?
  +--yes:attr_skew_all(A,N,O),O>3.64 ?
    +--yes:classvalue_freq(A,P,Q),Q>0.318 ?
      +--yes:classvalue_freq(A,L,R),R<0.097 ?
        +--yes:attr_gfunction(A,S,T),T>-0.437 ?
          +--yes:attr_relevance(A,S,1),safe(U>0.235) ?
            +--yes:lin_indiscr < mlcnb < RBFN < mlcib1 < MLP < ripper < ltree < c50r < c50b < c50t
            +--no: mlcnb < lin_indiscr < RBFN < MLP < ltree < mlcib1 < ripper < c50r < c50t < c50b
          +--no: classvalue_freq(A,L,V),V<0.003 ?
            +--yes:RBFN < c50b < MLP < lin_indiscr < mlcnb < ltree < mlcib1 < c50r < c50t < ripper
            +--no: RBFN < c50b < lin_indiscr < mlcnb < MLP < mlcib1 < ripper < ltree < c50r < c50t
            +--no: MLP < RBFN < mlcnb < lin_indiscr < mlcib1 < ripper < c50b < c50r < c50t < ltree
        +--no: attr_skew_all(A,N,W),W<5.217 ?
          +--yes:MLP < c50b < mlcnb < ripper < RBFN < c50t < lin_indiscr < c50r < ltree < mlcib1
          +--no: mlcnb < mlcib1 < RBFN < MLP < c50t < ltree < lin_indiscr < c50r < ripper < c50b
        +--no: classvalue_freq(A,X,Y),Y>0.786 ?
          +--yes:mlcnb < lin_indiscr < mlcib1 < c50b < ripper < ltree < MLP < c50r < c50t < RBFN
          +--no: attr_skew_all(A,Z,A1),A1<-0.612 ?
            +--yes:lin_indiscr < ripper < c50t < mlcnb < c50r < RBFN < ltree < MLP < c50b < mlcib1
            +--no: attr_relevance(A,B1,C1),safe(C1<0.062) ?
              +--yes:RBFN < lin_indiscr < mlcib1 < ripper < ltree < c50r < MLP < mlcib1 < c50b
              +--no: c50b < lin_indiscr < RBFN < mlcnb < MLP < ripper < ltree < c50r < c50t < mlcib1
          +--no: attr_skew_all(A,D1,E1),E1>2.095 ?
            +--yes:attr_skew_all(A,F1,G1),G1<-0.26 ?
              +--yes:RBFN < MLP < mlcnb < ripper < lin_indiscr < mlcib1 < c50b < c50r < c50t < ltree
              +--no: mlcnb < RBFN < lin_indiscr < mlcib1 < c50t < ltree < ripper < c50r < MLP < c50b
            +--no: attr_skew_all(A,H1,I1),I1<-1.303 ?
              +--yes:attr_kurt_all(A,H1,J1),safe(J1<1.631) ?
                +--yes:mlcib1 < c50t < MLP < c50r < ripper < RBFN < ltree < c50b < mlcnb < lin_indiscr
                +--no: mlcib1 < c50t < ripper < c50r < mlcnb < c50b < RBFN < ltree < lin_indiscr < MLP
              +--no: MLP < RBFN < mlcib1 < lin_indiscr < ltree < c50r < c50t < ripper < mlcnb < c50b
calcofluor_white = resistance (2.3%) (miss: 26.9%)
+-yes: notn = 1 (57.3%) (miss: 0%)
  |       +-yes: size: 7.8, 5.1, 2
  |     [1: 0.5898, 1, 0.3292]
  |     [1/5: 0.4485, 1, 0.1311]
  |     [1/5/1: 0.2448, 0.5, 0.0915]
  |     [1/5/4: 0.1992, 0.5, 0.0383]
  |       ...
  |
  |       +-no: size: 5.8, 6, 5
  |     ...
  |     [9: 0.3355, 0.4, 0.0779]
  |     [9/1: 0.3317, 0.4, 0.0601]
  |     ...
  |
  |       +-no: size: 570.3, 718.8, 545
  |     [1: 0.325, 0.3211, 0.3292]
  |     ...

Theory Complexity: Example

Classification (mutagenesis)

- B1
- B2
- B3
- B4

Nodes

- Progol
- FOIL
- TILDE
Induction Times: Example

Mutagenesis

- Progol
- FOIL
- TILDE
Scalability w.r.t. Data Set Size

Running times (mutagenesis)

- TILDE-classic
- TILDE-LDS
Overall Conclusions

- “Predictive clustering trees” represents a very general approach to predictive induction
  - Classification, regression, clustering, ranking, ...
- First order decision trees have interesting properties w.r.t. expressiveness
- Induction of FOLDTs inherits efficiency, accurateness, ... of TDIDT
- Highly versatile relational data mining approach
Some Pointers

- **RDM book**
  - Ch. 5: ILP “companion” systems (incl. Tilde)
  - Ch. 6: first order decision trees (S-CART)
  - Ch. 10: “upgrading” approach

- **Articles with Kramer, Blockeel, ...**
  - Algorithms & implementations
  - Applications: QSAR, biodegradability, water quality, ranking, ...
  - Knobbe et al., PKDD 2000: away from logic, towards relational databases

- **http://www.cs.kuleuven.ac.be/~dtai/ACE/**