Beyond Attribute-Value Data Mining

Lutz Hamel
Dept. of Computer Science and Statistics
University of Rhode Island
What is Data Mining?

Data mining is the application of *machine learning* techniques to large databases in order to extract *hidden knowledge*.

(KDD – Knowledge Discovery in Databases)
What is Machine Learning?

*Programs* that get better with *experience* given a task and some *performance measure*.

Most common is *inductive learning*, that is learning from a set of positive and negative examples.

- Learning to classify customers
- Learning to recognize spoken words
- Learning to play board games
What is Knowledge?

- Structural descriptions of data (transparent)
  - If-then-else rules
  - Decision trees
  - First-order logic theories
- Models of data (non-transparent)
  - Neural networks
  - Clustering (self-organizing maps, k-Means)
  - Naive-Bayes classifiers
Data Mining Today

- Today’s data mining tools are “single-table” oriented – *attribute-value oriented*.

- Basic assumption is that objects of a particular problem domain can be represented by a *fixed set* of attributes.
Attribute-Value Data Mining: Classification

<table>
<thead>
<tr>
<th>ShapeLeft</th>
<th>SizeLeft</th>
<th>ColorLeft</th>
<th>ShapeRight</th>
<th>SizeRight</th>
<th>ColorRight</th>
<th>DiagramPosition</th>
</tr>
</thead>
<tbody>
<tr>
<td>triangle</td>
<td>large</td>
<td>white</td>
<td>circle</td>
<td>large</td>
<td>black</td>
<td>left</td>
</tr>
<tr>
<td>triangle</td>
<td>large</td>
<td>black</td>
<td>circle</td>
<td>small</td>
<td>black</td>
<td>left</td>
</tr>
<tr>
<td>circle</td>
<td>small</td>
<td>white</td>
<td>triangle</td>
<td>small</td>
<td>black</td>
<td>left</td>
</tr>
<tr>
<td>circle</td>
<td>small</td>
<td>white</td>
<td>circle</td>
<td>large</td>
<td>grey</td>
<td>right</td>
</tr>
<tr>
<td>triangle</td>
<td>large</td>
<td>black</td>
<td>circle</td>
<td>large</td>
<td>grey</td>
<td>right</td>
</tr>
<tr>
<td>circle</td>
<td>large</td>
<td>black</td>
<td>triangle</td>
<td>small</td>
<td>white</td>
<td>right</td>
</tr>
</tbody>
</table>
Attribute-Value Data Mining: Classification

Given:

- A data universe $X$, here

$$X = \text{Shape Left} \times \text{Size Left} \times \text{Color Left} \times \text{Shape Right} \times \text{Size Right} \times \text{Color Right}$$

- A sample set $S$, where $S \subseteq X$

- A classification function $c: X \rightarrow \{\text{true, false}\}$, here

$$\text{Diagram Position}: X \rightarrow \{\text{left, right}\}$$

- Labeled training examples $D$, where

$$D = \{(s, c(s)) | s \in S\}$$

Use $D$ to determine:

- A function or hypothesis $c'$ such that $c'(x) \approx c(x)$ for all $x \in X$. 
Attribute-Value Data Mining: Decision Trees

- In decision tree learning the hypothesis $c'$ is represented as a tree.

- We can view decision tree learning as a heuristic search over all possible decision trees for the “best” tree.

A decision tree for our diagram problem would look like this:

```
+-----------------+  +-----------------+  +-----------------+
| ColorRight      |  | black            |  | grey             |  | white            |
    +-----------------+  +-----------------+  +-----------------+
    | left            |  | right            |  | right            |
```

8
A More Complicated Problem Domain

Difficult to represent with a fixed set of attributes:

- The scenes do not contain fixed numbers of objects.
- No inherent order of the objects in the scenes – difficult to express relations between objects.

Even if one forces an attribute-value representation – lots of “null” values in the table and exponential explosion of attributes.
First-Order Equational Logic

Equational logic is the logic of substituting equals for equals with algebras as models and term rewriting as the operational semantics.

theory LIST is
  sort List .
  sort ListElement .
  subsort ListElement < List .

  op _,_ : ListElement List -> List .
  op length : List -> Int .

  var E : ListElement .
  var L : List .

  eq length(E) = 1 .
  eq length(E,L) = 1 + length(L) .
end
First-Order Equational Logic

theory DIAGRAMS is
...
  eq diagram{ size(c1,medium)
    size(c2,small)
    size(t1,small) pointing(t1,up)
    size(t2,medium) pointing(t2,up)
    in(t1,c1) } = left.

  eq diagram{ size(c1,small)
    size(t1,large) pointing(t1,up)
    in(c1,t1) } = right.
end
First-Order Equational Logic

• First-order equational logic allows us to describe the diagrams in a very natural way.

• We can easily capture all the important aspects of object existence, characteristics, and relationships.

Why choose Equational Logic as the Representation Language?

• Precise semantics.

• Logical reasoning capabilities.

• Well developed module and type systems.
Deductive vs. Inductive Logic

- In (deductive) logic we deduce specific facts from general theories.
- In inductive logic we induce general theories from specific facts.

\[
\text{Deduction} \quad \text{Theories} \quad \text{Facts} \quad \text{Induction}
\]
Inductive Equational Logic

- In inductive equational logic we induce equational theories (hypotheses) from equations which represent the facts.

- Inductive equational logic admits the use of domain theories or background knowledge.

⇒ Inductive equational logic allows us to generalize from given facts and background knowledge.

⇒ In this setting we can consider inductive reasoning in equational logic to be data mining over first-order structures.
Inductive Equational Logic

Given:

- An observation universe $O$, here
  
  $$O = \{d \mid d \text{ is a left or right diagram description}\}.$$  

- A fact theory $F$, where $F \subseteq O$.

- A (possibly empty) background theory $B$.

Use $F$ and $B$ to determine hypothesis $H$:

- Use the relation $H \cup B \vdash f$, for all $f \in F$, to estimate an $H$ such that $H \cup B \vdash o$ for all $o \in O$.

- $H \cup B \vdash f$ means fact $f$ is *derivable* from $H$ and $B$.  
  

Inducing a Hypothesis

theory DIAGRAM-HYPOTHESIS is

...  
  eq diagram{ D } = left  
    if outside-is-circle(get-in(D)) and  
      inside-is-triangle(get-in(D)) .

  eq diagram{ D } = right  
    if outside-is-triangle(get-in(D)) and  
      inside-is-circle(get-in(D)) .

end

NOTE: the "helper functions" are elements of background theory $B$. 

16
Implementation of Inductive Equational Logic

- Determining the hypothesis $H$ can be considered a search over all possible hypotheses for the “best” hypothesis.
- Typically, the “best” hypothesis is the shortest theory from which all the facts in $F$ can be derived the theory that “explains” all the facts.
- We have implemented an experimental inductive equational logic programming system which utilizes evolutionary search techniques to search the hypotheses space for the “best” hypothesis.
- Evolutionary algorithms perform global searches rather than local, greedy searches, this results in very stable search results in the presence of noise in the fact theories.
- On the down side, evolutionary searches tend to be slow.
Mining Program Observations

- An interesting application of this technology is the mining of program observations or tests.
- Given a set of observations, we can construct a hypothesis that describes the behavior of the program in very concise terms.
- In general, it would be extremely difficult to capture program behavior in an attribute-value approach.

⇒ Reverse Engineering
⇒ Software Testing
Example: Even Predicate

Find a hypothesis describing the behavior of the predicate even:

theory EVEN-OBSERVATIONS is

    sort Int.
    op 0 : -> Int.
    op s : Int -> Int.
    op even : Int -> Bool.

    eq even(0) = true.
    eq even(s(s(0))) = true.
    eq even(s(s(s(s(0))))) = true.
    eq even(s(0)) = false.
    eq even(s(s(s(0)))) = false.
    eq even(s(s(s(s(s(0)))))) = false.

end
Example: Even Predicate

Hypothesis:

theory EVEN is

    sort Int .
    op 0 : -> Int .
    op s : Int -> Int .
    op even : Int -> Bool .
    var X : Int .

    eq even(s(s(X))) = even(X) .
    eq even(0) = true .

end
Example: Stack

Find a hypothesis describing the behavior of a stack module:

theory STACK-OBSERVATIONS is

  sorts Stack Element .
  ops a b c d: -> Element .
  op v : -> Stack .
  op top : Stack -> Element .
  op pop : Stack -> Stack .
  op push : Stack Element -> Stack .

  eq top(push(v,a)) = a .
  eq top(push(push(v,a),b)) = b .
  eq top(push(push(v,b),a)) = a .
  eq top(push(push(v,d),c)) = c .
  eq pop(push(v,a))= v .
  eq pop(push(push(v,a),b)) = push(v,a) .
  eq pop(push(push(v,b),a)) = push(v,b) .
  eq pop(push(push(v,d),c)) = push(v,d) .

end
Example: Stack

Hypothesis:

definition STACK is

  sorts Stack Element.
  op top : Stack -> Element.
  op pop : Stack -> Stack.
  op push : Stack Element -> Stack.
  var S : Stack.
  var E : Element.

  eq top(push(S,E)) = E.
  eq pop(push(S,E)) = S.

definition
Summary

- Today’s attribute-value data mining tools cannot capture the rich structure inherent in some interesting problem domains.
- Moving from an attribute-value representation to a first-order representation solves many of these representation problems.
- Equational logic is particularly well suited as a representation language due to its concise semantics and its well developed module and type systems.
- Our current, experimental implementation of inductive equational logic uses evolutionary search techniques and tends to be robust even in the presence of noise.
- Next steps include the move to a more efficient implementation based on C++ and the investigation of some large real-world problems.
Relevant Publications


