Incremental Rule Learning
with Partial Instance Memory
for Changing Concepts

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057-1232
maloof@cs.georgetown.edu
http://www.cs.georgetown.edu/~maloof

International Joint Conference on Neural Networks
Portland, OR
23 July 2003

Talk Overview

• Basics of rule learning
• On-line learning
• Concept memory and instance memory
• Partial instance memory and selecting examples
• Widmer and Kubat’s Window Adjustment Heuristic
• Evaluation on the STAGGER Concepts
The Basics of Rule Learning

The AQ algorithm (Michalski, 1969)

1. Start with positive and negative training examples
2. Pick one positive training example
3. Form a rule by generalizing it as much as possible without “covering” a negative example
4. Remove the positive examples covered by the rule
5. Until covering all positive examples, goto Step 2
6. Repeat for the negative class, goto Step 2

Learning the Concept of “Who can vote”

- Attributes:
  - gender ∈ { M, F }
  - age ∈ { 1, \ldots, 120 }
- Training examples:

<table>
<thead>
<tr>
<th>gender</th>
<th>age</th>
<th>vote</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>54</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>42</td>
<td>yes</td>
</tr>
<tr>
<td>M</td>
<td>22</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>32</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>11</td>
<td>no</td>
</tr>
<tr>
<td>M</td>
<td>14</td>
<td>no</td>
</tr>
<tr>
<td>M</td>
<td>8</td>
<td>no</td>
</tr>
<tr>
<td>F</td>
<td>16</td>
<td>no</td>
</tr>
</tbody>
</table>
Learning the Concept of “Who can vote”

- Pick a positive example: \( \langle M, 22, \text{yes} \rangle \)
- Rule: vote \( \leftarrow [\text{gender} = M] \& [\text{age} = 22] \)
- Generalize gender: vote \( \leftarrow [\text{gender} = M \lor F] \& [\text{age} = 22] \)
- Cover any negative examples? No!
- Generalize age: vote \( \leftarrow [\text{gender} = M \lor F] \& [\text{age} \geq 22] \)
- Cover any negative examples? No!
- Generalize age: vote \( \leftarrow [\text{gender} = M \lor F] \& [\text{age} > 16] \)
- Cover any negative examples? No!
- Final rule: vote \( \leftarrow [\text{age} > 16] \)

On-line Learning

- Training examples distributed over time
- But system must always be able to perform
- Temporal-Batch Learning
  1. Learn rules from examples
  2. Store rules, store examples
  3. Use rules to predict, navigate, etc.
  4. When new examples arrive, add to current examples
  5. Goto step 1
- Incremental Learning
  1. Learn rules from examples
  2. Store rules, discard examples
  3. Use rules to predict, navigate, etc.
  4. When new examples arrive, learn new rules using old rules and new instances
  5. Goto step 2
**Slide 7**

**Concept Memory**

- Full: Learner stores concept descriptions, changing them only when new examples arrive (e.g., WINNOW)
- No: Learner stores no concept descriptions that generalize training examples (e.g., IB2)
- Partial: Learner stores concept descriptions and modifies them but not necessarily in response to the arrival of new training examples, like weight decay (e.g., FAVORIT)

**Slide 8**

**Instance Memory**

- Full: Learner stores all examples from the input stream (e.g., ID5, GEM)
- No: Learner stores no examples (e.g., ID4, AQ11)
- Partial: Learner stores *some* examples (e.g., LAIR, HILLARY, FLORA, DARLING, METAL(B), METAL(IB), AQ-PM, AQ11-PM)
Classification of Learning Systems

On-line Learning Systems
- no concept memory
- partial concept memory
  - IB2
  - partial instance memory
  - full instance memory
  - IB1
- full concept memory

Algorithm for Learning with Partial Instance Memory

1. Learn rules from training examples
2. Select a portion of the examples
3. Store rules, store examples
4. Use rules to predict, navigate, etc.
5. When new examples arrive
   - if incremental learning, then
     - learn new rules using old rules, new instances, and examples held in partial memory
   - if temporal-batch learning, then
     - learn new rules using new instances and examples held in partial memory
6. Combine new instances with those in partial memory
7. Goto step 2
Selecting Examples for Partial Memory

- LAIR: the first positive example only
- HILLARY: only the negative examples
- DARLING: examples near the centers of clusters
- IB2: misclassified examples
- METAL(B), METAL(IB): sequence over a fixed window of time
- FLORA: sequence over a changing window, set adaptively
- AQ-PM, AQ11-PM, GEM-PM, AQ11-PM-WAH: examples on the boundaries of rules (i.e., extreme examples), possibly over a fixed window of time
  - for the rule: vote ← [age > 16]
  - extreme examples: ⟨F, 16, No⟩ and ⟨M, 22, Yes⟩
  - mark the boundary between the concepts ‘Can Vote’ and ‘Cannot Vote’

Visualization of Training Examples: Discrete Version of the Iris Data Set
**Induced Characteristic Rules**

\[
\text{setosa} \leftarrow [\text{pl} = 0] \& [\text{pw} = 0] \& \\
[\text{sl} = 0..3] \& [\text{sw} = 0, 2..4]
\]

\[
\text{versicolor} \leftarrow [\text{pl} = 1] \& [\text{pw} = 1..2] \& \\
[\text{sl} = 1..6] \& [\text{sw} = 0..4]
\]

**Visualization of Induced Rules**

Slide 14
Visualization of Extreme Examples

Evaluation of Learning Systems

- AQ11: Incrementally learns disjunctive rules. No instance memory. A lesioned version of AQ11-PM.
- AQ11-PM: Incrementally learns disjunctive rules. Selects examples on the boundaries of these descriptions over a fixed window of time.
- AQ11-PM-WAH: AQ11-PM with a dynamically sized window of time using Widmer and Kubat’s Window Adjustment Heuristic (WAH).
- AQ-PM: Temporal-batch learner. Disjunctive rules. Selects examples on the boundaries of these descriptions over a fixed window of time.
Widmer and Kubat’s WAH

Window-Adjustment-Heuristic()

- lc: threshold for low coverage, user-defined
- hc: threshold for high coverage, user-defined
- p: threshold for acceptable accuracy, user-defined
- N: examples covered by the positive concept description
- S: number of conditions in the positive description
- Acc: accuracy of current concept descriptions
- w: window size

if \( \frac{N}{S} < lc \lor (Acc < p \land \text{decreasing}(Acc)) \)
\[ \Delta w = -0.2w; \]
else if \( \frac{N}{S} > 2.0 \times hc \land Acc > p \)
\[ \Delta w = -1.0; \]
else if \( \frac{N}{S} > hc \land Acc > p \)
\[ \Delta w = 0.0; \]
else
\[ \Delta w = 1.0; \]
\[ w = w + \Delta w; \]
end.

The STAGGER Concepts

(size = small) \& (shape = circle) \lor (size = medium, large)

(color = red) \& (color = green)

a. Target concept for time steps 1–39.
b. Target concept for time steps 40–79.
c. Target concept for time steps 80–120.
Slide 19

The STAGGER Concepts:
Predictive Accuracy

![Graph showing predictive accuracy over time for different AQ models](image1)

Slide 20

The STAGGER Concepts:
Memory Requirements

![Graph showing memory requirements over time for different AQ models](image2)

(aq11 stores no examples.)
Observations

- Learners with partial instance memory (e.g., AQ11-PM-WAH, FLORA2) tend to outperform learners with no instance memory (e.g., STAGGER, AQ11)
- If the partial-memory learner can store the “right” examples, then when concepts change, it will have more examples for learning the new concept than will a learner that does not store examples
- Naturally, the trick is identifying the right examples
- Adding Widmer and Kubat’s WAH to AQ11-PM slightly improved accuracy
- But it did improve AQ11-PM’s generality since there’s no fixed period for storing examples
- On the other hand, instead of one parameter for the length of the fixed window, we now have three parameters for the WAH

Current and Future Work

- Better characterization of performance using synthetic data sets: CNF, DNF, m-of-n, class noise, concept overlap
- Scale to larger data sets: Just acquired 10 GB of audit data
- Track changing concepts in real data sets
- Evaluate effect of skewed data
- Prove bounds for predictive accuracy and examples maintained
- Reinforcement learning to determine WAH parameters
- Ensemble methods for concept drift (Kolter & Maloof, 2003)
Incremental Rule Learning
with Partial Instance Memory
for Changing Concepts

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC
maloof@cs.georgetown.edu
http://www.cs.georgetown.edu/~maloof

International Joint Conference on Neural Networks
Portland, OR
23 July 2003
References


