On-line Learning with Partial Instance Memory

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

Based on joint work with:
Ryszard Michalski (GMU & Polish Academy of Science)

IJCAI-01 Workshop on Learning from Temporal and Spatial Data
Seattle, WA
6 August 2001

Talk Overview

- On-line learning, concept memory, instance memory
- Algorithm for learning with partial instance memory
- Selecting extreme examples, those on the boundaries of concept descriptions
- Evaluation and Comparison
  - Computer Intrusion Detection
  - The STAGGER Concepts
- Future Work
On-line Learning
- Training examples distributed over time
- But system must always be able to perform
- Temporal-Batch Learning
  1. Learn, say, 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

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)
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, META(L(B), META(L(IB), AQ-PM, AQ11-PM)

Classification of Learning Systems

On-line Learning Systems
- IB2: partial instance memory
- IB1: full instance memory
- DARLING (AQ-PM)
- AQ-15C
- CN2
- C4.5
- ID4
- STAGGER
- WINNOW
- AQ11
- LAIR
- HILLARY
- FLORA2, 3, 4
- Meta(L(B), Meta(L(IB)
- AQ11-PM, GEM-PM

Copyright © 2001 Marcus A. Maloof
Algorithm for Learning with Partial 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: examples on the boundaries of rules (i.e., extreme examples), possibly over a fixed window of time
Visualization of Training Examples:
Discrete Version of the Iris Data Set

Induced Characteristic Rules

\[
\begin{align*}
\text{setosa} & \leftarrow [pl = 0] & [pw = 0] & [sl = 0..3] & [sw = 0, 2..4] \\
\text{versicolor} & \leftarrow [pl = 1] & [pw = 1..2] & [sl = 1..6] & [sw = 0..4]
\end{align*}
\]
Visualization of Induced Rules

Slide 11

Visualization of Extreme Examples

Slide 12
Evaluation of Learning Systems

- IB2: Instance-based learner. Selects misclassified examples
- FLORA2: Incrementally learns disjunctive rules. Selects examples over a window of time. Heuristic adjusts window size
- AQ11: Incrementally learns disjunctive rules. No instance memory. A lesioned version of AQ11-PM. Pascal implementation
- AQ11-PM: Incrementally learns disjunctive rules. Selects examples on the boundaries of these descriptions over a fixed window of time. Wrapper implementation
- AQ-PM: Temporal-batch learner. Disjunctive rules. Selects examples on the boundaries of these descriptions over a fixed window of time. C implementation

Computer Intrusion Detection

- Learning behavioral profiles of computing use for detecting intruders (also misuse)
- Derived our data set from the UNIX acctcom command
- Three weeks, over 11,200 records, selected 9 of 32 users
- Segmented into sessions: logouts and 20 minutes of idle time
- For each session, computed minimum, average, and maximum for seven numeric metrics
- Selected 10 most relevant: maximum real time, average and maximum system and user time, average and maximum characters transferred, average blocks read and written, maximum CPU factor, average hog factor
- Divided data into 10 partitions, used 1 for testing, 9 for training, applied methods, and repeated 30 times
Computer Intrusion Detection: Predictive Accuracy

- AQ11-PM
- AQ-PM
- AQ-BL
- AQ11
- IB2

Predictive Accuracy (%) vs. Time Step (t)

Computer Intrusion Detection: Memory Requirements

Examples Maintained vs. Time Step (t)

(aq11 stores no examples.)
Computer Intrusion Detection: Learning Times

![Graph showing learning times for different intrusion detection methods.](slide17_graph.png)

The STAGGER Concepts

- **(size = small)**
- **(shape = circle)**
- **(color = red)**

&

- **(size = medium, large)**
- **(color = green)**

### Slide 18

- **a.** Target concept for time steps 1–39.
- **b.** Target concept for time steps 40–79.
- **c.** Target concept for time steps 80–120.
The STAGGER Concepts:
Predictive Accuracy

The STAGGER Concepts:
Memory Requirements

(aq11 stores no examples.)
 Observations

- For static concepts, partial-memory learners, as compared to lesioned versions, tend to:
  - decrease predictive accuracy—often slightly
  - decrease memory requirements—often significantly
  - decrease learning time—often significantly
  - can decrease concept complexity
  - has little effect on performance time
- For changing concepts,
  - track concepts better than incremental learners with no instance memory (e.g., STAGGER, AQ11)
  - AQ11-PM tracks concepts comparably to FLORA2

 Future Work

- Better characterization of performance using synthetic data sets: CNF, DNF, m-of-n, class noise, concept overlap
- Scale to larger data sets: Acquiring more audit data
- Evaluate effect of skewed data: Rooftop detection
- Prove bounds for predictive accuracy and examples maintained
- Heuristics to adapt size of forgetting window
References


