Partial-Memory Learning
for Static and Changing Concepts

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

Based on work with Ryszard Michalski, GMU
Intelligent Systems Division
National Institute of Standards and Technology
Gaithersburg, MD
28 November 2001

Talk Overview

• Brief overview of machine learning (27%)
• Main topics (50%):
  – learning with partial instance memory
  – static and changing concepts
  – application to intrusion detection
• Other projects with other people:
  – machine learning to improve BUDDS, a vision system that
detects buildings in overhead imagery (20%)
  – Analysis of competing classifiers using components of
  variance of ROC measures (0%)
• Project on the horizon... (3%)
Learning from Examples

- One way humans (and computers) learn is from examples
- Imagine a child learning the concept ‘dog’

| spaniel | dog | + |
| husky   | dog | + |
| retriever | dog | + ← positive example |
| cat     | not a dog | - ← negative example |
| cow     | not a dog | - |
| wolf    | not a dog | - |
| boxer   | dog | + |

What is Being Learned?

- How does the child know it’s a dog?
- What features does the child use to recognize the dog? Its shape? color? fur? sound?
- What is the child learning?
  - learning the features that are predictive of dogs? “This animal has fur and barks, so it is a doggie”
  - remembering specific cases? “This animal sounds more like Lassie than Garfield, so it’s a doggie”
  - is she doing a little of both?
  - should machines do a little of both?
Testing and Generalization

- How do we know the child has learned ‘dog’?
  - Show her a poodle Child: dog Correct
  - Show her a lion Child: not a dog Correct
  - Show her a hyena Child: dog Oops, incorrect

- So we have the notions of *training* and *testing*
  - overtraining: performs well on the training examples, performs poorly on the testing examples
- We also have the notion of *generalization*:
  - she correctly identified the poodle and the lion but had never seen them before
  - over-generalization: everything is a dog!
  - under-generalization: nothing is a dog!

Accuracy and Error Costs

- By counting mistakes, we can measure accuracy:
  - true positive: saying ‘doggie’ to Lassie
  - true negative: saying ‘not a doggie’ to Garfield
  - false positive: saying ‘doggie’ to a hyena
  - false negative: saying ‘not a doggie’ to a doberman

- How should performance change with more and more training?
  - hopefully it increases! (unless we overtrain)
- Mistakes have different costs:
  - saying ‘not a doggie’ to a poodle: low cost
  - saying ‘doggie’ to a grizzly bear: HIGH COST!
Evaluation Methodology

- Like parents with children, ML researchers want to show their learning method is best!
- Or find the best method for an application
- How to do this in an unbiased way?
  - run experiments
  - IMPORTANT: Train on a randomly selected portion of the data and test on the remainder
  - plot accuracy: errors, types of errors
  - examine trade-offs
  - plot learning curves (i.e., accuracy over time)
  - plot accuracy at different decision thresholds (ROC analysis)
- Other performance measures: time, space, understandability of learned concepts

ROC Analysis

- ROC \equiv Receiver Operating Characteristic
- Lets us evaluate performance for a variety of error costs
- ROC curve plots the true positive and false positive rates at various decision thresholds
- The point (0, 1) is where classification is perfect, so we want curves that “push” toward this corner
- Traditional ROC analysis uses area under the curve as the measure of performance
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, ..., 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: $\langle M, 22, yes \rangle$
- 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 $\leftarrow [\text{gender} = M \lor F] \& [\text{age} > 22]$
- 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
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, METAL(B), METAL(IB), AQ-PM, AQ11-PM)
Classification of Learning Systems

On-line Learning Systems
- no concept memory
- partial concept memory
- full concept memory

- IB2
- FAVORIT
- DARLING
- AQ-PM
- AQ11
- AQ-15c
- CN2
- C4.5
- STAGGER
- Winnow
- AQ11
- ID4
- LAIR
- HILLARY
- FLORA2, 3, 4
- MetaL(B)
- MetaL(IB)
- AQ11-PM, GEM-PM

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: 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

Slide 18
Induced Characteristic Rules

setosa ← [pl = 0] & [pw = 0] & [sl = 0..3] & [sw = 0, 2..4]


Visualization of Induced Rules
Visualization of Extreme Examples

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: Memory Requirements

(aq11 stores no examples.)

The STAGGER Concepts

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

Slide 27

The STAGGER Concepts: Memory Requirements

Slide 28

(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

Current and Future Work: Partial-Memory Learning

- 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
- Heuristics to adapt size of forgetting window
Machine Learning to Improve BUDDS,
A Vision System that Detects Buildings
in Overhead Imagery

Joint work with:
Pat Langley (ISLE & Stanford)
Tom Binford (Stanford)
Ram Nevatia (USC)

Sponsors: DARPA through ONR, Sun Microsystems

Opportunities for Learning with BUDDS

- Lin and Nevatia (1996) present one approach to detecting buildings in RADIUS images; BUDDS uses knowledge at a number of levels:
  1. Grouping pixels into edge elements (i.e., edgels)
  2. Grouping edgels into lines
  3. Finding junctions and parallel lines
  4. Combining junctions and parallels into ‘Us’
  5. Grouping ‘Us’ into parallelograms (rooftop candidates)
  6. Verifying rooftop candidates (walls, shadows, overlap)
  7. Generating 3D building descriptions
- Learning can occur at any of these levels, but we focused on rooftop detection (step 5)
Attributes for Representing Rooftop Candidates

- BUDDS uses nine continuous attributes to evaluate rooftop candidates:
  1. Support for corners
  2. Support for parallel lines
  3. Support for orthogonal trihedral vertices
  4. Support for corner shadows
  5. Gaps in the edges of the candidate
  6. Displacement of edge support
  7. Lines crossing the candidate
  8. Existence of adjacent L-junctions and T-junctions

- We included each of these features in the training and test descriptions used for learning

Highlights of the Study

- Six images of Fort Hood, TX.
- Different locations, different aspects (nadir and oblique)
- Built a labeling tool that draws candidate rooftops on images
- Unequal and unknown error costs; highly skewed data set
- ROC analysis to compare classifiers
- Learning methods outperformed handcrafted classifier
- Evaluated generalization across location and aspect (Maloof, Langley, Binford, & Nevatia, 1998)
- User studies (Ali, Langley, Maloof, Sage, & Binford, 1998)
- Investigated multi-level learning (Maloof, 2000)
Visualization Interface for Labeling Rooftop Candidates

Slide 35

ROC Curve for All Image Data

Slide 36

Copyright © 2001 Marcus A. Maloof
Areas Under the ROC Curves

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Area under ROC Curve</th>
</tr>
</thead>
<tbody>
<tr>
<td>C5.0</td>
<td>0.867±0.006</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>0.854±0.009</td>
</tr>
<tr>
<td>Perceptron</td>
<td>0.853±0.010</td>
</tr>
<tr>
<td>k-NN (k = 11)</td>
<td>0.847±0.006</td>
</tr>
<tr>
<td>BUDDS Classifier</td>
<td>0.802±0.014</td>
</tr>
</tbody>
</table>

ANOVA: $p < 0.01$, LabMRMC: $p < 0.001^*$

* The current implementation of LabMRMC is limited to five treatments (i.e., learning algorithms), so we conducted this analysis for the best five and not all twelve.

Project on the Horizon...

- Security in Ad hoc Networks
  - Levine and Fagg (UMass), Royer and Almeroth (UCSB), Maloof and Shields (Georgetown)
  - proposed to National Science Foundation
  - machine learning for anomaly detection
References


