OIL
Online Incremental Learning

Fabrice Harel-Canada
Migyeong (Mimi) Gwak
Roadmap

● Conceptual Overview
  ○ What is it
  ○ Why is it important
  ○ Applications
  ○ Key Challenges
  ○ Models

● Examples
  ○ Learn++
  ○ ADAIN
What is OIL?

- Continuous model adaptation (i.e. learning) from streaming data that arrives over time
- Formally, an OIL system must do all of the following:
  - Learns from new data, generally as soon as it becomes available
  - Does not require access to previously processed data (one pass)
  - Preserves previously acquired knowledge
  - Accommodates new classes
Why is OIL important?

- Handles limited upfront data
  - Expensive or time consuming
  - Unknown size or distribution
- Lower memory consumption
  - Doesn’t need to store previously seen data for reprocessing
- Foundation for greater system autonomy
Applications

- Big Data analytics
- Robotics
- Self-Driving Transport
- Image/Video Processing
- Automated Annotation
- Outlier Detection
What are the key challenges?

- Concept drift
- Stability-plasticity dilemma
- Adaptive model complexity
- Efficient memory representation
- Model benchmarking
Concept Drift

Changes in the distribution of data over time

- What is changing?
  - The input distribution ($\mathbf{x}$) → virtual / covariate drift
  - The function itself $p(y|\mathbf{x})$ → real drift

- How fast is the data changing?
  - Slow → drift
  - Fast → shift

- How widespread is the data changing?
  - Only some areas → localized drift
  - Everything → systemic drift
The Stability-Plasticity Dilemma

- Learning new things tends to overwrite previously acquired knowledge.
- Update too quickly and previously encoded data will be forgotten constantly, sometimes catastrophically.
- Update too slowly and impaired adaptation hurts performance.
- Moderation between these two is key!
Adaptive Model Complexity

- Unknown data means that model complexity and meta-params cannot be specified upfront
- Reduces critical meta-params like learning rate into another model parameter to be tuned automatically
Efficient Memory Representation

- Compact representations of knowledge
  - Invariants
    - E.g. classification error for a drift detector
  - Implicit representations params
    - Most common
  - Explicit
    - E.g. limited set of typical examples (prototypes / exemplars)
Model Benchmarking

- Incremental vs Non-incremental
  - Task accuracy of the models using data from the same timeframe
  - The one-pass nature of most stream processing is a critical constraint in these comparisons
- Incremental vs Incremental
  - Measures of robustness and classification error against concept drift
Types of OIL Models

- Support Vector Machines (SVM)
- Connectionist Models
  - Multilayer perceptrons (MLP) / Neural Networks
- Input Partitioners
  - Trees / windows
- Prototype-based
- Ensembles
  - A little bit of everything
Learn++: An Incremental Learning Algorithm for Supervised Neural Networks

Robi Polikar | Lalita Udpa | Satish S. Udpa | Vasant Honavar

IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS VOL. 31, NO. 4, NOVEMBER 2001
Learn++

• Early example of an ensemble approach to incremental learning for classification tasks
• Builds upon the concept of boosting “weak learner” into “strong learner” by combining them together
• Uses weighted majority voting between these boosted classifiers to make the final output class prediction
• Focuses on novelty by selecting more strongly for incorrectly classified data
Incremental learning via weighted weak classifiers
How does Learn++ overcome the challenges of OIL?

- It handles concept drift by focusing on hard / novel data
- It handles the stability-plasticity dilemma by using weighted majority voting so that new classes can be accommodated when encountered, while still strongly weighting the learners that were good at classifying the previous set of classes
- It handles adaptive model complexity by adding new classifiers until performance can no longer be improved
- It compactly represents knowledge in params / weights
Concentric Circles

- 2 attributes \((x, y)\)
- 5 classes
- Goal
  - Initialize the OIL with data from the 1, 2, 3
  - Start including examples from new classes
## Results

### Training and Generalization Performance of Learn++ on Concentric Circles Database

<table>
<thead>
<tr>
<th>Inc. Train → Dataset</th>
<th>Training 1 (10)</th>
<th>Training 2 (10)</th>
<th>Training 3 (13)</th>
<th>Training 4 (3)</th>
<th>Training 5 (15)</th>
<th>Training 6 (7)</th>
<th>Last 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>98.7%</td>
<td>96.7%</td>
<td>91.4%</td>
<td>91.4%</td>
<td>95.3%</td>
<td>95.3%</td>
<td>41.7%</td>
</tr>
<tr>
<td>$S_2$</td>
<td>---</td>
<td>96.1%</td>
<td>87.1%</td>
<td>85.8%</td>
<td>92.2%</td>
<td>91.6%</td>
<td>40.6%</td>
</tr>
<tr>
<td>$S_3$</td>
<td>---</td>
<td>---</td>
<td>98.3%</td>
<td>98.3%</td>
<td>72%</td>
<td>90.8%</td>
<td>51.5%</td>
</tr>
<tr>
<td>$S_4$</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>93.6%</td>
<td>77%</td>
<td>88.4%</td>
<td>49.8%</td>
</tr>
<tr>
<td>$S_5$</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>88%</td>
<td>95.2%</td>
<td>60.4%</td>
</tr>
<tr>
<td>$S_6$</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>96.4%</td>
<td>53.6%</td>
</tr>
<tr>
<td><strong>TEST</strong></td>
<td><strong>55.6%</strong></td>
<td><strong>56.8%</strong></td>
<td><strong>73.2%</strong></td>
<td><strong>74.4%</strong></td>
<td><strong>85.8%</strong></td>
<td><strong>89.6%</strong></td>
<td><strong>52.8%</strong></td>
</tr>
</tbody>
</table>

- Test shows monotonic improvement as it learns how to classify unknown classes
- Large accuracy improvements coincide with new class introductions in 3 and 5
- Last 7 highlights the importance of previously learned classifications
Incremental Learning from Stream Data

Haibo He | Sheng Chen | Kang Li | Xin Xu

IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 22, NO. 12, DECEMBER 2011
A general **ADAptive INcremental** learning framework.

Motivated by the adaptive boosting principle and ensemble learning methodology.

Learning from stream data for classification.

Accumulating experience over time.

Using such knowledge to improve future learning and prediction performance.

Different base classifiers can be integrated.
System Architecture
Learning Procedure

1. Estimate the initial distribution function for the current data set \((D_t)\) using a mapping function.
   \[
   \hat{P}_{t-1} = \varphi(D_{t-1}, D_t, P_{t-1})
   \]

2. Apply previous hypothesis \((h_{t-1})\) to \(D_t\) calculate the pseudo-error of \(h_{t-1}\)
   \[
   \varepsilon_{t-1} = \sum_{j: h_{t-1}(x_j) \neq y_j} \hat{P}_{t-1}(j)
   \]

3. Update the distribution \((P_t)\) for \(D_t\)
   \[
   P_t(j) = \frac{\hat{P}_{t-1}(j)}{Z_t} \times \begin{cases} 
   \beta_{t-1} & \text{if } h_{t-1}(x_j) = y_j \\
   1 & \text{otherwise}
   \end{cases}
   \]

4. Develop a hypothesis \((h_t)\) based on \(D_t\) with \(P_t\)

5. Repeat the procedure when the next chunk of data set \((D_{t+1})\) is received.
Mapping Function

- Transforms the knowledge from the current data chunk into the learning process of the future data chunks
- Provides a connection from past experience to the newly received data
Simulation Results

<table>
<thead>
<tr>
<th>Data set</th>
<th>ADAIN.SVR</th>
<th>Learn++</th>
<th>ADAIN.SVR</th>
<th>Learn++</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>training</td>
<td>testing</td>
<td>training</td>
<td>testing</td>
</tr>
<tr>
<td>Spambase</td>
<td>128.42</td>
<td>2.19</td>
<td>600.26</td>
<td>246.62</td>
</tr>
<tr>
<td>Magic</td>
<td>7256.18</td>
<td>14.36</td>
<td>107256.33</td>
<td>1623.74</td>
</tr>
<tr>
<td>Waveform</td>
<td>163.51</td>
<td>3.39</td>
<td>548.84</td>
<td>238.1016</td>
</tr>
<tr>
<td>Sat</td>
<td>352.31</td>
<td>4.95</td>
<td>1246.62</td>
<td>346.53</td>
</tr>
<tr>
<td>Data sets</td>
<td>Methods</td>
<td>Prediction accuracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
<td>---------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>class 1</td>
<td>class 2</td>
<td>class 3</td>
</tr>
<tr>
<td>Spambase</td>
<td>ADAIN.MLP</td>
<td>0.8820</td>
<td>0.9352</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>ADAIN.SVR</td>
<td>0.8990</td>
<td>0.9205</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>IMORL</td>
<td>0.9106</td>
<td>0.8929</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Accumulation</td>
<td>0.8803</td>
<td>0.9190</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Learn++</td>
<td>0.8532</td>
<td>0.9561</td>
<td>--</td>
</tr>
<tr>
<td>Magic</td>
<td>ADAIN.MLP</td>
<td>0.9315</td>
<td>0.7137</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>ADAIN.SVR</td>
<td>0.9319</td>
<td>0.7395</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>IMORL</td>
<td>0.8404</td>
<td>0.7836</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Accumulation</td>
<td>0.8670</td>
<td>0.7410</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Learn++</td>
<td>0.9523</td>
<td>0.6786</td>
<td>--</td>
</tr>
<tr>
<td>Waveform</td>
<td>ADAIN.MLP</td>
<td>0.7843</td>
<td>0.8230</td>
<td>0.8193</td>
</tr>
<tr>
<td></td>
<td>ADAIN.SVR</td>
<td>0.7576</td>
<td>0.8198</td>
<td>0.8474</td>
</tr>
<tr>
<td></td>
<td>IMORL</td>
<td>0.7575</td>
<td>0.8000</td>
<td>0.8009</td>
</tr>
<tr>
<td></td>
<td>Accumulation</td>
<td>0.7070</td>
<td>0.7558</td>
<td>0.7534</td>
</tr>
<tr>
<td></td>
<td>Learn++</td>
<td>0.7870</td>
<td>0.8360</td>
<td>0.9072</td>
</tr>
<tr>
<td>Sat</td>
<td>ADAIN.MLP</td>
<td>0.9602</td>
<td>0.9131</td>
<td>0.9169</td>
</tr>
<tr>
<td></td>
<td>ADAIN.SVR</td>
<td>0.9584</td>
<td>0.8889</td>
<td>0.9305</td>
</tr>
<tr>
<td></td>
<td>IMORL</td>
<td>0.9000</td>
<td>0.8918</td>
<td>0.8566</td>
</tr>
<tr>
<td></td>
<td>Accumulation</td>
<td>0.9452</td>
<td>0.9473</td>
<td>0.8697</td>
</tr>
<tr>
<td></td>
<td>Learn++</td>
<td>0.9696</td>
<td>0.8860</td>
<td>0.9327</td>
</tr>
<tr>
<td>Data set</td>
<td>ADAIN.MLP</td>
<td>BoostOnline</td>
<td>VW</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------------</td>
<td>-----</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OA</td>
<td>AUC</td>
<td>OA</td>
<td>AUC</td>
</tr>
<tr>
<td>Spambase</td>
<td>0.9302</td>
<td>0.8777</td>
<td>0.9003</td>
<td>0.8709</td>
</tr>
<tr>
<td>Magic</td>
<td>0.8553</td>
<td>0.9100</td>
<td>0.7448</td>
<td>0.8056</td>
</tr>
<tr>
<td>Waveform</td>
<td>0.8530</td>
<td>0.9649</td>
<td>0.7393</td>
<td>0.8949</td>
</tr>
<tr>
<td>Sat</td>
<td>0.8484</td>
<td>0.9781</td>
<td>0.7211</td>
<td>0.9393</td>
</tr>
</tbody>
</table>
How does ADAIN overcome the challenges of OIL?

- It handles concept drift by focusing on hard / novel data
- It compactly represents knowledge in a single hypothesis
- It handles the stability-plasticity dilemma by setting incrementally developed hypotheses and pruning the obsolete hypothesis to better keep tuned on the evolving data stream
- It handles adaptive model complexity by the mapping function
<table>
<thead>
<tr>
<th><strong>Learn++</strong></th>
<th><strong>ADAIN</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>● Use weighted majority voting between these boosted classifiers</td>
<td>● Obtain hypothesis and distribution function from the current chunk of data and improves upon it</td>
</tr>
<tr>
<td>● Add new classifiers until performance can no longer be improved</td>
<td>● Use mapping function to obtain weights of instances in a new chunks of the data</td>
</tr>
</tbody>
</table>
QUESTIONS?
1. Alexander Gepperth, Barbara Hammer. Incremental learning algorithms and applications. European Symposium on Artificial Neural Networks (ESANN), 2016, Bruges, Belgium.
