Multi-label kNN Classifier with Self Adjusting Memory for Drifting Data Streams

Martha Roseberry and Alberto Cano

Department of Computer Science
Virginia Commonwealth University
{mroseberry, acano}@vcu.edu
Outline

• Data Stream Mining and SAM-kNN
• Multi-label Classification
• Multi-label Self Adjusting Memory k Nearest Neighbors
• Experimental setup and multi-label metrics
• Results and discussion
• Conclusions and future work
Data Stream Mining

- A potentially infinite sequence of instances that arrive in rapid succession
- Limited amount of time to process each instance
- Never have access to all instances simultaneously
- Likely to evolve over time, experiencing concept drift
  - Change in the concept or distribution of data classes
  - Abrupt, gradual or incremental
  - Recurring
  - Mixed
Self Adjusting Memory k Nearest Neighbors

- Biologically inspired proposal by Losing et al.
- Uses both short- and long-term memories (STM and LTM), with a kNN classifier
- Most recent instances form the STM
  - Reduces in size when current concept changes
- Older instances form the LTM
  - Evenly compressed when a maximum size is reached
- Robust ability to adapt to varying concept drifts

Multi-label Classification

• Rather than being associated with a single class, each instance is associated with a set of classes
• Relevant to text classification, scene classification, etc.
• Predictions can be partially correct
• Imbalance is measured using cardinality and density
  – Cardinality is the average number of labels associated with an instance
  – Density is the cardinality divided by the number of labels
• Cardinality and density often both low, making much multi-label data highly imbalanced
• Few works address multi-label data streams
Multi-label kNN with Self Adjusting Memory

- ML-SAM-kNN: a classifier for multi-label data streams using a STM and LTM with a majority-vote kNN adapted for multi-label
ML-SAM-kNN: adapting the STM

• The short-term memory contains the most recent \( m \) instances.
• When an instance is added, different windows sizes are evaluated
  – Tested sizes are \( m, m/2, m/4 \) ...
• STM windows were evaluated using their Hamming scores

\[
\text{Hamming score} = \frac{1}{NL} \sum_{i=0}^{N} \sum_{l=0}^{L} 1 \mid y_l = z_l, y_l \in Y_i, z_l \in Z_i
\]

  – \( N = \) number of instances, \( L = \) number of labels, \( Y_i = \) true labeset, \( Z_i = \) predicted labelset
• Discarded instances are cleaned and transferred to the LTM
ML-SAM-kNN: cleaning and transfer

• To clean with respect to an instance \( s \) in the STM, for each label a threshold \( \theta_l \) is defined as:

\[
\theta_l = \max\{d(s, s_i) \mid s_i \in N_k(STM), y_l(s_i) = y_l(s)\}
\]

  – \( N_k(STM) \) is the k nearest neighbors of \( s \) in the STM

• In the set to be cleaned, instances closer than \( \theta_l \) without label \( l \) are removed

• Instances discarded from STM are cleaned with respect to each instance in the STM

• The LTM is cleaned with respect to each newly added instance
ML-SAM-kNN: compressing the LTM

• To preserve past concepts and infrequent data, instances in the LTM are not faded out with time
• Compressed when the size of the memories exceeds a maximum
• For each labelset, $|M_{LTY}|/2$ clusters are found, where $M_{LTY}$ is the set of instances in the LTM with labelset $Y$
• New instances with labelset $Y$ are created with the cluster centroids as the feature vectors
• After compression, LTM consists of these new instances, reducing the size by half
Multi-label kNN with Self Adjusting Memory

- For each instance, the STM, LTM and the CM (combined union of the two) each induces a multi-label kNN classifier. The classifier with the highest Hamming score is used for the final prediction.
Experimental Setup

Datasets: 23 multi-label datasets, including the cardinality and density

Algorithms:
- MLS – none
- BRU – learner: HoeffdingTree
- CCU – learner: HoeffdingTree
- PSU – learner: HoeffdingTree
- RTU – learner: HoeffdingTree
- BMLU – learner: HoeffdingTree, components: 10
- ISOUPT – default
- ML-kNN – k: 10, window: 1000
- ML-SAM-kNN – k:5, maxSTM: 400, maxLTM: 600

Software: MOA 2018.04

Hardware: Intel Xeon CPU E—2690v4 with 128 GB of memory
**Multi-label Evaluation Metrics**

\[
\text{Subset accuracy} = \frac{1}{N} \sum_{i=0}^{N} \mathbb{1} | Y_i = Z_i
\]

\[
\text{Precision} = \frac{1}{N} \sum_{i=0}^{N} \frac{|Y_i \cap Z_i|}{|Y_i|}
\]

\[
\text{Hamming score} = \frac{1}{NL} \sum_{i=0}^{N} \sum_{l=0}^{L} \mathbb{1}
\]

\[
y_l = z_l, \ y_l \in Y_i, \ z_l \in Z_i
\]

\[
\text{Recall} = \frac{1}{N} \sum_{i=0}^{N} \frac{|Y_i \cap Z_i|}{|Z_i|}
\]

\[
\text{Accuracy} = \frac{1}{N} \sum_{i=0}^{N} \frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}
\]

\[
\text{F-measure} = \frac{1}{N} \sum_{i=0}^{N} \frac{2 \times |Y_i \cap Z_i|}{|Y_i| + |Z_i|}
\]
Average results of the algorithms for all metrics

<table>
<thead>
<tr>
<th>Average</th>
<th>MLS</th>
<th>BRU</th>
<th>CCU</th>
<th>PSU</th>
<th>RTU</th>
<th>BMLU</th>
<th>ISOUPT</th>
<th>ML-kNN</th>
<th>ML-SAM-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset accuracy</td>
<td>0.2007</td>
<td>0.2426</td>
<td>0.2558</td>
<td>0.2008</td>
<td>0.2034</td>
<td>0.2435</td>
<td>0.1810</td>
<td>0.2792</td>
<td><strong>0.3284</strong></td>
</tr>
<tr>
<td>Hamming score</td>
<td>0.8787</td>
<td>0.9105</td>
<td>0.9113</td>
<td>0.8842</td>
<td>0.8936</td>
<td>0.9116</td>
<td>0.9084</td>
<td>0.9192</td>
<td><strong>0.9224</strong></td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.1574</td>
<td>0.2586</td>
<td>0.2636</td>
<td>0.2195</td>
<td>0.1380</td>
<td>0.2513</td>
<td>0.1602</td>
<td>0.2889</td>
<td><strong>0.3478</strong></td>
</tr>
<tr>
<td>Precision</td>
<td>0.1760</td>
<td>0.2999</td>
<td>0.3018</td>
<td>0.2430</td>
<td>0.1382</td>
<td>0.2918</td>
<td>0.1786</td>
<td>0.3182</td>
<td><strong>0.3845</strong></td>
</tr>
<tr>
<td>Recall</td>
<td>0.2103</td>
<td>0.3412</td>
<td>0.3424</td>
<td>0.2927</td>
<td>0.1977</td>
<td>0.3313</td>
<td>0.2391</td>
<td>0.3788</td>
<td><strong>0.4350</strong></td>
</tr>
<tr>
<td>F-Measure</td>
<td>0.1840</td>
<td>0.3021</td>
<td>0.3049</td>
<td>0.2539</td>
<td>0.1544</td>
<td>0.2933</td>
<td>0.1926</td>
<td>0.3298</td>
<td><strong>0.3915</strong></td>
</tr>
<tr>
<td>Evaluation time (s)</td>
<td><strong>64</strong></td>
<td>10071</td>
<td>12008</td>
<td>95</td>
<td>2404</td>
<td>25445</td>
<td>17184</td>
<td>17404</td>
<td>16394</td>
</tr>
<tr>
<td>Model cost (RAM-Hours)</td>
<td><strong>2.0E-5</strong></td>
<td>8.5E-1</td>
<td>1.4E+0</td>
<td>2.5E-4</td>
<td>1.3E-1</td>
<td>2.7E+0</td>
<td>2.1E+0</td>
<td>4.9E-1</td>
<td><strong>8.8E-1</strong></td>
</tr>
</tbody>
</table>

- ML-SAM-kNN obtains the best average results for all six quality metrics.
- High computational complexity, but it achieves runtimes comparable to BRU, CCU, ISOUPT, and ML-kNN and is faster than BMLU
Bonferroni-Dunn test

- Indicates only ML-kNN and CCU cannot be claimed as significantly different, the others being worse for all or some quality metrics
Wilcoxon test for all metrics ($p$-values)

<table>
<thead>
<tr>
<th>ML-SAM-kNN vs.</th>
<th>MLS</th>
<th>BRU</th>
<th>CCU</th>
<th>PSU</th>
<th>RTU</th>
<th>BMLU</th>
<th>ISOUPT</th>
<th>ML-kNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subset accuracy</td>
<td>7.34E-4</td>
<td>7.34E-4</td>
<td>2.30E-2</td>
<td>5.41E-3</td>
<td>5.79E-4</td>
<td>4.28E-3</td>
<td>6.03E-5</td>
<td>3.84E-2</td>
</tr>
<tr>
<td>Hamming score</td>
<td>6.41E-5</td>
<td>1.36E-2</td>
<td>3.65E-2</td>
<td>7.87E-6</td>
<td>7.41E-5</td>
<td>1.73E-2</td>
<td>2.42E-2</td>
<td>1.00E00</td>
</tr>
<tr>
<td>Accuracy</td>
<td>4.95E-5</td>
<td>1.94E-2</td>
<td>1.78E-2</td>
<td>1.03E-2</td>
<td>2.77E-4</td>
<td>3.48E-3</td>
<td>2.77E-4</td>
<td>4.41E-2</td>
</tr>
<tr>
<td>Precision</td>
<td>4.30E-5</td>
<td>1.36E-2</td>
<td>2.12E-2</td>
<td>1.13E-2</td>
<td>2.15E-4</td>
<td>3.86E-3</td>
<td>2.15E-4</td>
<td>2.51E-2</td>
</tr>
<tr>
<td>Recall</td>
<td>4.30E-5</td>
<td>1.36E-2</td>
<td>8.55E-3</td>
<td>8.55E-3</td>
<td>1.64E-3</td>
<td>4.74E-3</td>
<td>2.04E-3</td>
<td>2.42E-1</td>
</tr>
<tr>
<td>F-Measure</td>
<td>4.30E-5</td>
<td>1.13E-2</td>
<td>1.24E-2</td>
<td>8.55E-3</td>
<td>3.55E-4</td>
<td>2.82E-3</td>
<td>2.77E-4</td>
<td>3.77E-2</td>
</tr>
<tr>
<td>Evaluation time (s)</td>
<td>2.38E-7</td>
<td>6.65E-1</td>
<td>6.87E-1</td>
<td>4.77E-7</td>
<td>9.15E-3</td>
<td>2.59E-1</td>
<td>2.12E-2</td>
<td>5.22E-2</td>
</tr>
<tr>
<td>Model cost (RAM-Hours)</td>
<td>2.38E-7</td>
<td>2.54E-2</td>
<td>2.33E-2</td>
<td>2.38E-7</td>
<td>6.03E-5</td>
<td>7.49E-2</td>
<td>5.01E-1</td>
<td>9.41E-1</td>
</tr>
</tbody>
</table>

- A $p$-value < 0.05 indicates significant difference
- Significant differences between ML-SAM-kNN and ML-kNN for all quality metrics except Hamming score and recall
- Significant difference between ML-SAM-kNN and CCU for all
Subset accuracy and F-Measure (Mediamill dataset)
Performance of the STM, LTM & CM (Mediamill dataset)

Subset accuracy vs instances for different methods:
- ML-SAM-kNN
- ML-kNN
- STM
- LTM
- CM

F-Measure vs instances for different methods:
- ML-SAM-kNN
- ML-kNN
- STM
- LTM
- CM
Conclusions

• ML-SAM-kNN performs demonstrably better than its peers, exhibiting statistically better results for nearly all metrics.
• Performance comes at the expense of runtime, but ML-SAM-kNN performs comparably with the top ranked algorithms.
• Poor performance of the LTM indicates that the compression and clustering are not appropriate for multi-label data.
• Need to address this issue, explicitly handle label imbalance and adjust memory architecture to account for label dependencies.
Multi-label kNN Classifier with Self Adjusting Memory for Drifting Data Streams

Martha Roseberry and Alberto Cano

Department of Computer Science
Virginia Commonwealth University
{mroseberry, acano}@vcu.edu