Adaptive XML Tree Mining on Evolving Data Streams

Albert Bifet  Ricard Gavaldà

Laboratory for Relational Algorithmics, Complexity and Learning LARCA
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

ECML-PKDD 2009
Bled, september 8th, 2009
Pattern Mining in Data Streams

Our setting

- Patterns: Objects where
  
  
  \( p \) subpattern of \( q \) "makes sense (partial order)"
  
- Sets, sequences, trees, graphs

- Mining frequent patterns

- Classifying patterns

- In a data stream that changes over time
<?xml version="1.0"?>
<?xml−stylesheet type="text/xsl" href="movie1.xsl"?>
<filmlibrary>
   <name>Classic Films</name>
   <movie>
      <title>The Bicycle Thief</title>
      <genre>Social Drama</genre>
      <language>Italian</language>
      <year>1948</year>
      <length>90 Min.</length>
      <filmtype>BW</filmtype>
      <credits>
         <director>Vittorio de Sica</director>
         <story>Gennarino Bartolini</story>
         <cinematography>Carlo Montuori</cinematography>
      </credits>
   </movie>
</filmlibrary>
Issues

Pattern Classification
- Mapping patterns → features
- Frequent patterns, closed patterns, generators, ...

Data stream classification
- Highly sublinear memory (in #items seen)
- Low processing time per item
- Tolerate distribution & concept change
To our knowledge, first tree pattern classifier in data streams
Builds features by mining frequent closed subtrees or maximal subtrees
  - Miner is interesting in itself / competitive with state-of-the-art
Uses recently proposed ensemble methods for classification
Implementation over the MOA framework
Tree (Pattern) Mining

Task occurs in chemistry, computer vision, text retrieval, bioinformatics, Web analysis, XML queries, . . .

- A transaction *supports* a tree if the tree is a subtree of the transaction
- *Support* of a tree is the number of transactions that support it

Given a dataset of trees and value *min_support*, find

- Frequent Tree mining (FT):
  all trees whose support is no less than *min_support*
- Closed Frequent Tree mining (CT):
  + no super-tree with the same support
Previous Work

- [Zaki-Agrawal] Classifier from frequent sets + Bayesian rules
- [Kudo et al.] Classifier from “significant” frequent trees + boosting
- [Collins et al., Kashima et al.] SVM’s, tree kernels; feature space = frequent trees
- CMTreeMiner [Chi et al.], [Termier et al.] Dryade: Closed frequent tree miners *without* computing all frequent trees
- [Li et al 06] Frequent subtree miner for XML data streams
- [Bifet-G 08] Frequent closed pattern miner in data streams, unlabelled trees
Closure Operator on Trees

- $\mathcal{D}$: the finite input dataset of trees
- $\mathcal{T}$: the (infinite) set of all trees

**Definition**

We define the following Galois connection pair:

- For finite $A \subseteq \mathcal{D}$
  - $\sigma(A)$ is the set of subtrees of the $A$ trees in $\mathcal{T}$
    \[
    \sigma(A) = \{ t \in \mathcal{T} \mid \forall t' \in A \left( t \preceq t' \right) \}
    \]

- For finite $B \subset \mathcal{T}$
  - $\tau_{\mathcal{D}}(B)$ is the set of supertrees of the $B$ trees in $\mathcal{D}$
    \[
    \tau_{\mathcal{D}}(B) = \{ t' \in \mathcal{D} \mid \forall t \in B \left( t \preceq t' \right) \}
    \]
Closure Operator

The composition $C_D = \sigma \circ \tau_D$ is a closure operator.

Characterizing closed trees

A tree $t$ is closed (no supertree with same support) in $D$ iff

$C_D(t) = \{t\}$
Rules for adding and removing patterns to datasets [Bifet-G 08]:

**Theorem**

Let $D_1$ and $D_2$ be two datasets of patterns. A pattern $t$ is closed for $D_1 \cup D_2$ if and only if

- $t$ is a closed pattern for $D_1$, or
- $t$ is a closed pattern for $D_2$, or
- $t$ is a subpattern of a closed pattern in $D_1$ and of a closed pattern in $D_2$ and $C_{D_1 \cup D_2}(\{t\}) = \{t\}$.

**Theorem**

Let $D$ be a pattern dataset. A pattern $t$ is closed for $D$ if and only if the intersection of all its closed superpatterns is $t$. 
Incremental Algorithm

Computing the lattice of frequent trees

Construct empty lattice $L$;

Repeat

Collect batch of $B$ trees;

Build closed tree lattice for $B$, $L_2$;

$L := \text{merge}(L, L_2)$ (using addition rule)

Memory & time depend on lattice size (number of closed trees) not on DB size!

Efficient ops. using the representation for trees by [Balcázar-Bifet-Lozano]
Incremental Algorithm

Computing the lattice of frequent trees

Construct empty lattice \( L \);

Repeat

Collect batch of \( B \) trees;

Build closed tree lattice for \( B, L_2 \);

\( L := \text{merge}(L, L_2) \) (using addition rule)

Memory & time depend on lattice size (number of closed trees) not on DB size!

Efficient ops. using the representation for trees by [Balcázar-Bifet-Lozano]
Dealing with time changes

- Keep a window on recent stream elements
  - Actually, just its lattice of closed sets!
- Keep track of number of closed trees in lattice, \( N \)
- Use some change detector on \( N \)
- When change is detected:
  - Drop stale part of the window
  - Update lattice to reflect this deletion, using deletion rule

Alternatively, sliding window of some fixed size
Miner is interesting in itself

Can also be used for static databases
For small number of labels:
  - slightly faster than CMTreeMiner
  - significantly less memory than CMTreeMiner
  - (CMTreeMiner keeps all dataset in memory)

T8M synthetic dataset [Zaki02]:
100 labels, mother tree size 10,000, DBsize 8M
A tree is maximal if no supertree of $t$ is frequent
All maximal trees are closed

- Non-maximal closed patterns can be derived from maximal ones
- ...but not their supports
- Are they still enough for classification?
XML Tree Classification on evolving data streams

Figure: A dataset example
XML Tree Classification on evolving data streams

<table>
<thead>
<tr>
<th>Id Tree</th>
<th>Closed Trees</th>
<th>Maximal Trees</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1 0 1</td>
<td>1 1 0</td>
<td>CLASS1</td>
</tr>
<tr>
<td>2</td>
<td>0 0 1 1</td>
<td>0 0 1</td>
<td>CLASS2</td>
</tr>
<tr>
<td>3</td>
<td>1 0 1 1</td>
<td>1 0 1</td>
<td>CLASS1</td>
</tr>
<tr>
<td>4</td>
<td>0 1 1 1</td>
<td>0 1 1</td>
<td>CLASS2</td>
</tr>
</tbody>
</table>

The table above shows the classification of XML tree patterns into closed and maximal trees, along with their corresponding classes.
XML Tree Framework on evolving data streams

Two components:

- An XML closed frequent tree miner
- A Data stream classifier algorithm, which we will feed with tuples to be classified online.

Attributes in these tuples represent the occurrence of the current closed trees in the originating tree, although the classifier algorithm need not be aware of this.
WEKA: the bird
MOA: the bird

The Moa (another native NZ bird) is not only flightless, like the Weka, but also extinct.
Massive Online Analysis is a framework for online learning from data streams.

http://www.cs.waikato.ac.nz/~abifet/MOA/

- It is closely related to WEKA
- It includes a collection of offline and online algorithms and tools for evaluation:
  - Hoeffding Trees, Hoeffding option trees
  - Boosting and bagging. In particular:
    - Adaptive-Size Hoeffding Tree bagging & boosting [Bifet et al., KDD09]
  - with and without Naïve Bayes classifiers at the leaves.
Experiments: Synthetic datasets

- Zaki’s tree dataset generator
- 2 mother trees, 2 classes, depth and fanout 10
- 1M samples, node labels change every 250,000 trees

<table>
<thead>
<tr>
<th>Bagging</th>
<th>Time</th>
<th>Acc.</th>
<th>Mem.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaTreeMiner</td>
<td>161.61</td>
<td><strong>80.06</strong></td>
<td>4.93</td>
</tr>
<tr>
<td>IncTreeMiner</td>
<td>212.75</td>
<td>65.73</td>
<td>4.4</td>
</tr>
</tbody>
</table>
Experiments: Real dataset

LOGML files [Zaki 02] describing 3 weeks of user sessions logs, each as XML file classes = .edu vs. non-.edu visitors

<table>
<thead>
<tr>
<th>Dataset</th>
<th># Trees</th>
<th>Maximal</th>
<th></th>
<th>Closed</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLOG12</td>
<td>15483</td>
<td>84</td>
<td>79.64</td>
<td>1.2</td>
<td>228</td>
</tr>
<tr>
<td>CSLOG23</td>
<td>15037</td>
<td>88</td>
<td>79.81</td>
<td>1.21</td>
<td>243</td>
</tr>
<tr>
<td>CSLOG31</td>
<td>15702</td>
<td>86</td>
<td>79.94</td>
<td>1.25</td>
<td>243</td>
</tr>
<tr>
<td>CSLOG123</td>
<td>23111</td>
<td>84</td>
<td>80.02</td>
<td>1.7</td>
<td>228</td>
</tr>
</tbody>
</table>
Conclusions

- A tree / XML tree stream classifier system
- Frequent closed / maximal trees as features
- Frequent closed tree miner based on closure operators
- That reacts quickly to distribution / label changes
- Maximal trees may suffice
Future Work

- More experiments for better understanding of behavior
- Especially, comparison with CMTreeMiner
- Deletion of obsolete attributes
- Use generators instead of closed / maximal
- XML mining in data streams when #labels is large