Adaptive XML Tree Mining on Evolving Data Streams

Albert Bifet ABIFET@LSI.UPC.EDU
Ricard Gavaldà GAVALDA@LSI.UPC.EDU
Universitat Politècnica de Catalunya, Departament de Llenguatges i Sistemes Informàtics

Abstract

We propose a new method to classify trees, using closed and maximal frequent trees. Closed trees maintain the same information as frequent trees using less space and maximal trees maintain approximate information. We use them to reduce the number of classification features. We present a new framework for data stream tree classification. For the first component of our classification framework, using a methodology based in Galois Lattice Theory, we present three closed tree mining algorithms: an incremental one IncTreeMiner, a sliding-window based one, WinTreeMiner, and finally one that mines closed trees adaptively from data streams, AdaTreeMiner. To the best of our knowledge this is the first work on tree classification in streaming data varying with time. We give a first experimental evaluation of the proposed classification method.

1. Introduction

Tree classification and the frequent tree discovery task have been important tasks over the last decade. Nowadays, they are becoming harder, as the size of the trees datasets is increasing and we cannot assume that data has been generated from a static distribution. If we want accuracy in the results of our algorithms, we have to consider that the distribution that generates data may vary over time, often in an unpredictable and drastic way.

Tree Mining is becoming an important field of research due to the fact that XML patterns are tree patterns and that XML is becoming a standard for information representation and exchange over the Internet. XML data is growing and it will soon constitute one of the largest collection of human knowledge. XML tree classification has been done traditionally using information retrieval techniques considering the labels of nodes as bags of words. With the development of frequent tree miners, classification methods using frequent trees appeared (Zaki & Aggarwal, 2003). Recently, closed frequent miners were proposed (Chi et al., 2001), and using them for classification tasks is the next natural step.

A longer version of this paper will be available from the first author webpage.

Presented at *ILP-MLG-SRL*, Leuven, Belgium, 2009.

2. Preliminaries

Trees are connected acyclic graphs, *rooted trees* are trees with a vertex singled out as the root, and *unranked* trees are trees with unbounded arity. We say that $t_1, \ldots, t_k$ are the *components* of tree t if t is made of a node (the root) joined to the roots of all the t_i's. We can distinguish betweeen the cases where the components at each node form a sequence (ordered trees) or just a set (*unordered trees*). We will deal with rooted, unranked trees.

An *induced subtree* of a tree t is any connected subgraph rooted at some node v of t that its vertices and edges are subsets of those of t. An *embedded subtree* of a tree t is any connected subgraph rooted at some node v of t that does not break the ancestor-descendant relationship among the vertices of t. We are interested in induced subtrees.

The (infinite) set of all trees will be denoted with $\mathcal{T}$, but actually all our developments will proceed in some finite subset of $\mathcal{T}$ which will act as our universe of discourse.

The input to our data mining process, now is a given finite or infinite dataset $\mathcal{D}$ of transactions, where each transaction $s \in \mathcal{D}$ consists of a transaction identifier, *tid*, a tree, and a discrete class label. Tids are supposed to run sequentially from 1 to the size of $\mathcal{D}$. From

that dataset, our universe of discourse $\mathcal{U}$ is the set of all trees that appear as subtree of some tree in $\mathcal{D}$.

Figure 1 shows a finite dataset example of trees.

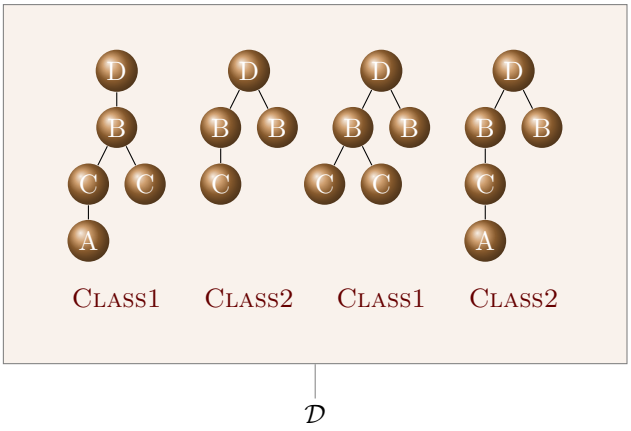


Figure 1. A dataset example

Following standard usage, we say that a transaction s *supports* a tree t if t is a subtree of the tree in transaction s. The number of transactions in the dataset $\mathcal{D}$ that support t is called the *support* of the tree t. A subtree t is called *frequent* if its support is greater than or equal to a given threshold *min_sup*. The frequent subtree mining problem is to find all frequent subtrees in a given dataset. Any subtree of a frequent tree is also frequent and, therefore, any supertree of a nonfrequent tree is also nonfrequent (the *antimonotonicity* property).

2.1. Frequent Tree Compression

We define a frequent tree t to be *closed* if none of its proper supertrees has the same support as it has. Generally, there are much fewer closed trees than frequent ones. In fact, we can obtain all frequent subtrees with their support from the set of frequent closed subtrees with their supports. So, the set of frequent closed subtrees maintains the same information as the set of all frequent subtrees.

We define a frequent tree t to be *maximal* if none of t's proper supertrees is frequent. All maximal trees are closed, but not all closed trees are maximal, so there are more closed trees than maximal. Note that we can obtain all frequent subtrees without their support from the set of maximal frequent subtrees. So, the set of maximal frequent subtrees maintains approximately the same information as the set of all frequent subtrees.

Maximal trees are trees that do not have any frequent supertree. All maximal trees are closed trees.

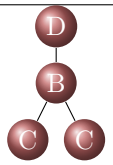
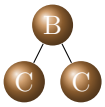
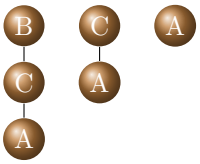
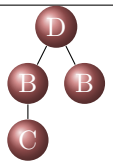
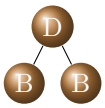
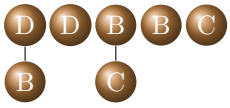
	Closed	Freq. not Closed Trees	Tree Trans. 1	2	3	4
c_1			1	0	1	0
c_2			1	0	0	1
c_3			0	1	1	1
c_4			1	1	1	1

Figure 2. Frequent trees from dataset example (*min_sup* = 30%), and their corresponding attribute vectors.

If *min_sup* is zero, then maximal trees are the transaction trees.

Following standard usage on Galois lattices, we consider now implications of the form $A \rightarrow B$ for sets of trees A and B from $\mathcal{U}$. Specifically, we consider the following set of rules: $A \rightarrow \Gamma_{\mathcal{D}}(A)$. Alternatively, we can split the consequents into $\{A \rightarrow t \mid t \in \Gamma_{\mathcal{D}}(A)\}$.

It is easy to see that $\mathcal{D}$ obeys all these rules: for each A, any tree of $\mathcal{D}$ that has as subtrees all the trees of A has also as subtrees all the trees of $\Gamma_{\mathcal{D}}(A)$.

Proposition 1 *Let t_i be a frequent tree for $\mathcal{D}$. A transaction tree t satisfies $t_i \preceq t$, if and only if it satisfies $\Delta_{\mathcal{D}}(t_i) \preceq t$.*

We use Proposition 1 to reduce the number of attributes on our classification task, using only closed frequent trees, as they keep the same information. The attribute vector of a frequent tree will be the same as its closed tree attribute vector. Also, we may reduce the number of attributes on our classification task, using only maximal frequent trees, as they keep approximately the same information as closed frequent trees.

BAGGING		Maximal						Closed					
		Unordered			Ordered			Unordered			Ordered		
	# Trees	Att.	Acc.	Mem.	Att.	Acc.	Mem.	Att.	Acc.	Mem.	Att.	Acc.	Mem.
CSLOG12	15483	84	**79.64**	1.2	77	79.63	1.1	228	78.12	2.54	183	78.12	2.03
CSLOG23	15037	88	**79.81**	1.21	80	79.8	1.09	243	78.77	2.75	196	78.89	2.21
CSLOG31	15702	86	**79.94**	1.25	80	79.87	1.17	243	77.6	2.73	196	77.59	2.19
CSLOG123	23111	84	**80.02**	1.7	78	79.97	1.58	228	78.91	4.18	181	78.91	3.31
BOOSTING		Maximal						Closed					
		Unordered			Ordered			Unordered			Ordered		
	#Trees	Att.	Acc.	Mem.	Att.	Acc.	Mem.	Att.	Acc.	Mem.	Att.	Acc.	Mem.
CSLOG12	15483	84	**79.46**	1.21	77	78.83	1.11	228	75.84	2.97	183	77.28	2.37
CSLOG23	15037	88	79.91	1.23	80	**80.24**	1.14	243	77.24	2.96	196	78.99	2.38
CSLOG31	15702	86	**79.77**	1.25	80	79.69	1.17	243	76.25	3.29	196	77.63	2.62
CSLOG123	23111	84	79.73	1.69	78	**80.03**	1.56	228	76.92	4.25	181	76.43	3.45

Table 1. Comparison of tree classification algorithms. Memory is measured in MB. The best individual accuracies are indicated in boldface (one per row).

3. XML Tree Classification framework on data streams

Our XML Tree Classification Framework has two components:

- An XML closed frequent tree miner, for which we could use any incremental algorithm that maintains a set of closed frequent trees.
- 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.

For the first component of the framework, we propose three tree mining algorithms adapting the general framework for trees presented in (Bifet & Gavaldà, 2008), and extending unlabelled tree mining methods to labelled ones:

- IncTreeMiner, an incremental closed tree mining algorithm,
- WinTreeMiner, a sliding window closed tree mining algorithm
- AdaTreeMiner, an adaptive closed tree mining algorithm

The second component of the framework is based on MOA. **M**assive **O**nline **A**nalysis (MOA) (Holmes et al., 2007) is a framework for online learning from continuous supplies of examples, such as data streams. It is closely related to the well-known WEKA project, and it includes a collection of offline and online as well as tools for evaluation. In particular, it implements boosting, bagging, and Hoeffding Trees, both with and without Naïve Bayes classifiers at the leaves.

4. Experimental evaluation

We evaluate our approach to tree classification on real classification data sets. The real CSLOG data set spans 3 weeks worth of such XML user-sessions. To convert this into a classification data set they chose to categorize each user-session into one of two class labels: edu corresponds to users from an "edu" domain, while other class corresponds to all users visiting the CS department from any other domain. They separate each week's logs into a different data set (CSLOGx, where x stands for the week; CSLOG12 is the combined data for weeks 1 and 2). Notice that the edu class has much lower frequency rate than other.

Table 1 shows the results on bagging and boosting using 10 Hoeffding Trees with adaptive Naive Bayes leaf predictions. Comparing maximal trees with closed trees, we see that maximal trees use 1/4 to 1/3rd of attributes, 1/3 of memory, and they perform better.

References

Bifet, A., & Gavaldà, R. (2008). Mining adaptively frequent closed unlabeled rooted trees in data streams. *KDD'08*.

Chi, Y., Xia, Y., Yang, Y., & Muntz, R. (2001). Mining closed and maximal frequent subtrees from databases of labeled rooted trees. *Fundamenta Informaticae*, *XXI*, 1001–1038.

Holmes, G., Kirkby, R., & Pfahringer, B. (2007). MOA: Massive Online Analysis. `http://sourceforge.net/projects/moa-datastream`.

Zaki, M. J., & Aggarwal, C. C. (2003). Xrules: an effective structural classifier for xml data. *KDD '03* (pp. 316–325). New York, NY, USA: ACM.