Mining Generalised Emerging Patterns

Xiaoyuan Qian, James Bailey, Christopher Leckie

Department of Computer Science and Software Engineering
University of Melbourne, Australia
{jbailey, caleckie}@csse.unimelb.edu.au

Abstract. Emerging Patterns (EPs) are a data mining model that is useful as a means of discovering distinctions inherently present amongst a collection of datasets. However, current EP mining algorithms do not handle attributes whose values are asscociated with taxonomies (*is-a* hierarchies). Current EP mining techniques are restricted to using only the leaf-level attribute-values in a taxonomy. In this paper, we formally introduce the problem of mining generalised emerging patterns. Given a large data set, where some attributes are hierarchical, we find emerging patterns that consist of items at any level of the taxonomies. Generalised EPs are more concise and interpretable when used to describe some distinctive characteristics of a class of data. They are also considered to be more expressive because they include items at higher levels of the hierarchies, which have larger supports than items at the leaf level. We formulate the problem of mining generalised EPs, and present an algorithm for this task. We demonstrate that the discovered generalised patterns, which contain items at higher levels in the hierarchies, have greater support than traditional leaf-level EPs according to our experimental results based on ten benchmark datasets.

1 Introduction

An important problem in data mining is how to characterise the differences between two data sets. A widely used approach to this problem is to find *emerging patterns* (EPs), which can be used to describe significant changes between two data sets [4]. EPs are conjunctions of simple conditions representing a particular class of records. Emerging patterns have strong discriminating power and are thus very useful for describing the contrasts that exist between two classes of data. A key advantage of emerging patterns is that since they are basically conjunctions of simple conditions, they are very easy to understand.

In many application domains, taxonomies (is-a hierarchies) are available for attribute values. For example, an attribute corresponding to a geographical location can be represented at many levels of detail, such as city, state or country. Until now, emerging patterns have only been capable of representing contrasts between datasets whose attributes are non-hierarchical. The patterns discovered have been at the lowest level of representation. This can lead to a large number of similar patterns being discovered, involving attribute values that are distinct, yet semantically related, e.g., cities that are all part of the same state. This

creates two problems. First the support of each such pattern in isolation will be less than the support of the group of related patterns. Second, a large group of similar patterns is more difficult for an expert to understand and verify, compared to a single, more general pattern. Therefore, our motivation has been to develop an algorithm to mine *generalised* emerging patterns, which has the capability of dealing with data sets whose attribute values are associated with *is-a* taxonomies, such that the contrasts discovered will contain information not only from the lowest-level but also from any other levels in the hierarchy.

In this paper, we introduce a new knowledge pattern, called *generalised emerging patterns*. With the introduction of the concept of generalised EPs, we now allow a much larger and semantically extended pattern base by associating membership hierarchies with some attributes in a dataset. Quantitative data such as age or income can have meaningful attribute hierarchies achieved by aggregating base values into increasing, non-overlapping ranges. For example, dates of birth can be generalised to month and year of birth.

A critical challenge in this problem is that attribute hierarchies greatly expand the size of the space of potential EPs that could be mined. Patterns can now express a mixed level of concept aggregation realized by using combinations of values across different levels in the hierarchy. These patterns have improved expressiveness, potential usefulness and understandability for decision makers over traditional EPs. The resultant multi-dimensional generalised EPs are more concise and interpretable. If we are given a large number of leaf-level EPs, we might prefer having fewer, but more expressive EPs that have stronger discriminating power when used to describe distinctive characteristics of a class of data objects. However, the pattern space is greatly expanded after including attribute values from higher levels in the hierarchy. This makes the efficient computation of generalised EPs an important research challenge.

There are three main contributions of this research. First, we introduce the concept of generalised emerging patterns and formulate the problem of mining generalised emerging patterns. Second, we develop an algorithm for mining these patterns, and analyse the behaviour of the mining method over a variety of real-life datasets. Third, we analyse the compactness and expressiveness of the discovered generalised EPs in comparison with traditional non-hierarchical EPs.

2 Background and Related Work

In this section we present basic definitions which are used throughout this paper. All definitions in this section are adapted from [7]. A dataset is a collection of data objects of the form $(a_1, a_2, ..., a_n)$ following the schema $(A_1, A_2, ..., A_n)$ where each of the objects is called an **instance** and $A_1, A_2, ..., A_n$ are called **attributes**. Each data object in the dataset is also labelled by a class label $C \in \{C_1, C_2, ..., C_k\}$ which indicates the class to which the data object belongs.

An item is a pair of the form (attribute-name, attribute-value). A set of items is called an **itemset** (or a **pattern**). We say any instance S contains an itemset X, if $X \subseteq S$. The **support** of an itemset X in a dataset D, denoted as

$sup_D(X)$, is $Count_D(X)/|D|$, where $Count_D(X)$ is the number of instances in D containing X, and $|D|$ is the total number of instances in D.

We follow the definition of Emerging Patterns used in [4],[3],[2],[6], also known as Jumping Emerging Patterns. An EP is an itemset whose support increases abruptly from zero in one data set (known as the negative data set), to non-zero in another data set (known as the positive data set) - the ratio of support-increase being infinite. An itemset X is said to be an **Emerging Pattern (EP)** from D_1 to D_2 if $sup_{D_1}(X) = 0$ and $sup_{D_2}(X) > 0$. An EP X is said to be **minimal** if there does not exist another EP Y such that $Y \subset X$.

A related field is the problem of mining association rules. The aim of mining association rules is to find all rules that satisfy a user-specified minimum *support* and minimum *confidence* [9],[11],[12]. The concept of *generalised* association rules first appeared in [10]. The problem of mining *generalised* association rules was defined informally as – given a set of transactions and a taxonomy, find association rules where the items may be from **any level** of the taxonomy. The approach taken by [10] was to first add all ancestors of each item in a transaction T, so that an extended transaction T' is obtained. After extending all transactions, any non-generalised association rule mining algorithm could then be applied to the extended transactions. Work presented in [5] further improved efficiency by imposing a lexicographic order on the itemsets such that rightmost Depth-First-Search could be used. In more recent work [8], "more-general-than" hierarchies were associated with *each* attribute in a large dataset. This is more closely related to our research, since in the context of mining emerging patterns, attributes are often multivalued and are independent from any other attributes in the data set. Their algorithm, **GenTree** [8], uses a tree structure which represents the multi-dimensional generalisation relations among all data tuples in a relational dataset over a set of hierarchical attributes.

Current techniques of mining EPs do *not* handle hierarchical attributes. Moreover, the techniques of dealing with taxonomy structures in the context of association rule mining cannot be applied directly to EP mining. A key difference is that while association rule learning requires search for high support rules in a single data set, EPs must satisfy the additional constraint of low support in the negative data set. This additional constraint creates the need for an efficient algorithm to prune candidate generalisations that match the negative data set. Therefore, the aim of our research is to address the problem of mining emerging patterns for data sets whose attributes are associated with hierarchies. This problem is formally described in the next section.

3 Generalised Emerging Patterns

The type of datasets we consider have a set of attributes, one or more of which is associated with a taxonomy which represents *is-a* relations on its attribute-values. Figure 1 shows an example of two such attribute taxonomies.

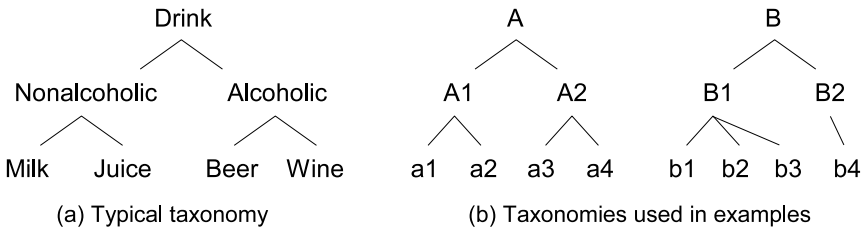

Fig. 1. Examples of taxonomic hierarchies

A **generalised emerging pattern** is an itemset whose support increases from zero in one class of data, to non-zero in another class, and consists of items from any level of the taxonomy associated with the corresponding attributes.

Given a data set D_1 of positive instances and data set D_2 of negative instances, there are many different EPs that can be found. An **EP space** is defined as the set of all EPs with repsect to D_1 and D_2 [9]. It has been shown that the EP space can be concisely represented by two bounds $\langle \mathcal{L}, \mathcal{R} \rangle$ [9]. The left bound $\mathcal{L}$ contains the *most general* EPs, and the right bound $\mathcal{R}$ contains the *most specific* EPs, where an itemset X is said to be more general than another itemset Y if $X \subset Y$.

Note that in the case of generalised EPs, the pattern space has been greatly expanded since items are not only restricted to leaves of the taxonomies. Our aim is to find the most general EPs from a class of data. The traditional approach is to take the collection from the left bound of the border representation because the concept of generality is equivalent to the concept of minimality (or most expressive). However, with the introduction of non-leaf-level items in the patterns, the word *general* here no longer implies *minimal* only, it also implies the *highest-level* possible in the hierarchy. Therefore an extended generalisation measure must be introduced to allow comparison of two patterns and decide whether or not one is considered more general than the other in both dimensions.

The existence of hierarchies on attribute-values now enables us to compare one item with another within the same attribute domain. We say that one item X is *more general* than another item Y if and only if $X \in ancestors(Y)$. In our small example above, item $A1$ is considered more general than $a2$. Considering the fact that hierarchies exist on individual attributes, only items from the same attribute are comparable. Therefore, only EPs consisting of items from the same combinations of attributes are are allowed to be compared with each other. For example, $\{A_1, c1\}$ and $\{b2, c1\}$ are not comparable because they have items from different attributes. However, $\{A_1, c1\}$ and $\{a3, c2\}$ *are* considered to be comparable. We refer to pattern $X = \{x_1, x_2, ..., x_m\}$ as a **generalisation** of pattern $Y = \{y_1, y_2, ..., y_m\}$ if $x_i \in ancestors(y_i)$ for at least one i ($i = 1$ to m); and $x_k = y_k$ for $k = 1$ to m where $k \neq i$. X is also said to be **more general** than Y. Y is referred to as a **specialisation** of (or **more specific** than) X.

3.1 Problem Description

Given a set of instances $\mathcal{D}$, our aim is to mine the most general set of generalised EPs which contain items from any level of the taxonomies *and* have infinite growth rate from one class to all others.

Considering a generalised pattern X, if any of its specialisations is supported by the positive instances, X itself is then supported by the positive instances. Likewise, if any of its specialisatons is supported by the negative instances, X itself is then supported by the negative instances. Therefore, all legal generalised emerging patterns must have the following properties: (1) At least **one** of its specialisations must have non-zero support in the *positive* class; (2) **None** of its specialisations should have non-zero support in the *negative* class.

The set of most general EPs that could be mined with respect to a set of positive instances and a set of negative instances should satisfiy the properties of **completeness** and **conciseness**.

- **Completeness**: The set of most general EPs mined, S, can be used as the left bound of the border representation such that $< \mathcal{S}, \mathcal{R} >$ represents all generalised EPs in a set.
- **Conciseness**: The set S is in its most concise form because any pattern which is either a subset or a generalisation of some EP in the set is not an EP anymore.

4 Algorithms for Mining Generalised EPs

Let us consider the problem of mining generalised EPs in terms of the problem of mining traditional EPs. A set of EPs mined from a dataset can be represented using a border representation $< \mathcal{L}, \mathcal{R} >$, where the left bound $\mathcal{L}$ is the set of minimal EPs, while the right bound $\mathcal{R}$ is the set of most specific EPs. Since we intend to discover the most general set of generalised EPs, we can therefore post-process on the left bound to obtain a more general set $\mathcal{L}'$ such that the new border $< \mathcal{L}', \mathcal{R} >$ represents the complete set of generalised emerging patterns in a dataset. In this generalised case, a pattern is considered an EP if:

- it is a superset *or* a specialisation *or* a specialisation of a superset of a certain EP in $\mathcal{L}'$ *and*
- it has a leaf-level specialisation that is a subset of an EP in $\mathcal{R}$

Hence, the problem of discovering the most general generalised EPs can be decomposed into three subproblems:

1. Mine the set of non-hierarchical EPs from the dataset using an existing mining algorithm
2. Enumerate all possible generalised patterns based on the set $\mathcal{L}$ of minimal EPs (the left bound)
3. Prune all generalised patterns that are not legal emerging patterns

Since existing EP mining techniques can be used for subproblem 1, our algorithm has been developed to address subproblems 2 and 3.

4.1 Algorithm GTree

Let us motivate our algorithm by first considering a brute-force approach to mining generalised EPs. The brute-force approach is to take each EP from the collection and enumerate all possible generalised patterns using ancestors of the items this EP contains. For each generalised pattern, check through the negative instances to see whether it is supported in the negative class and delete it if it has non-zero support in the negative class. All generalised patterns that do not have support in the negative class are considered as legal generalised EPs. Before each generalised EP is added to the final set, it must be checked against the EPs that are already in the set to make sure that it is removed if it is a specialisation of any existing patterns *or* the specialisations of it are removed if it is considered more general than any existing patterns.

We have added several important optimisations to the Brute-Force algorithm to develop an algorithm called **GTree**. In this algorithm, the enumeration of generalised patterns is now achieved through building a set enumeration tree for each input non-hierarchical EP. There are several important optimisations introduced by this approach. First, fewer passes are needed over the negative class data, since only one pass over the negative instances is required for each tree, instead of for each generalised pattern. In addition, taxonomic information can improve pruning efficiency. When an item is pruned, its corresponding generalisations are also pruned, since all generalisations will also be supported by the negative class. Let us now examine the GTree algorithm in detail.

4.2 GTree Construction

A GTree is a set-enumeration tree constructed from a set of leaf-level EPs in the left bound $\mathcal{L}$. The tree represents all possible generalised patterns that can be enumerated from the leaf-level EPs. The tree has a root, which does not contain any value. The height of the tree is the length of the EP. Each path from root to leaf represents one possible generalised pattern. There is an imposed ordering on the tree, such that paths from left to right are patterns from the most specific to the most general.

Before constructing the tree, we first divide the set of non-hierarchical EPs into a number of **groups** (G_1 to G_k) according to the combination of attributes in each pattern, such that each **group** should only contain patterns with items from the *same* combination of attributes. For each group, we then construct one tree for all EPs in that group. For example, consider a dataset where $\{a1, b2\}$ and $\{a1, b3\}$ are both non-hierarchical EPs. Each of these EPs will result in the same generalisation tree. Consequently, these patterns should be grouped together, so that a single tree can be constructed for both patterns, and then pruning on the negative patterns need occur only once.

The formal algorithm for constructing a tree is outlined as follows:

```
Given E = {input non-hierarchical minimal EPs}
for each group of EPs G_k ∈ L do
   Tree r ← buildTree( G_k )

buildTree( group G )
   initialise empty tree with root node r
   for each EP p_j ∈ G do
      insertPattern( r, p_j, 1 )
   return tree r;

Let p[l] denote the L^th item in pattern p
   e.g., if p = (a_1, b_1), p[1] = a_1, p[2] = b_1
assign ordering of values in attribute l and their generalisations,
most specific to most general

insertPattern( node n, pattern p, l )
   for each item i ∈ p[l] ∪ generalisations(p[l])
      if ( i ∉ n.children ) then
         insert i in n.children according to ordering of attribute l
      find node n_i ∈ n.children that corresponds to item i
      insertPattern( n_i, p, ++l );
```

As an example, consider a group of EPs from $\{a1, b1\}$, $\{a2, b2\}$, $\{a3, b3\}$, $\{a3, b4\}$, $\{a4, b3\}$. The resulting GTree is shown in Figure 2. We can see that items along each path from the root to the leaf forms a generalised itemset and paths from left to right are from the most specific to the most general.

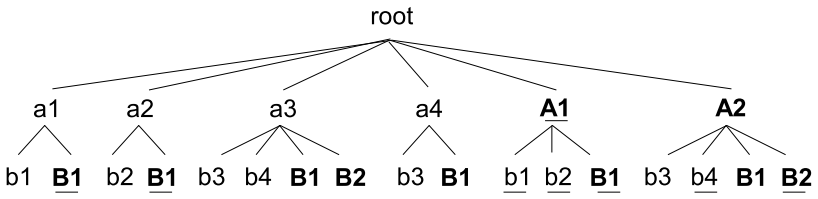

Fig. 2. A *GTree* built based on the example in Figure 1. Note that underlined nodes correspond to nodes that are pruned in the discussion in Section 4.3

4.3 Pruning from GTree

Pruning invalid EPs from the tree requires testing all negative instances one by one.

```
For each pattern p from the negative dataset
   pruneTree( root r, p, 1 )

pruneTree( node n, pattern p )
   for each item i ∈ pattern p
      if ( attribute type of item i ≠ attribute type of n.children ) then
         skip to next item i
      else
         for each node n_c ∈ n_c.children
            if ( n_c.children ≠ null ) then
               pruneTree( n_c, p )
            else delete n_c from n_c.children
```

For example, if the training dataset contains the negative instances $\{a1, b2\}$, $\{a1, b3\}$, $\{a2, b1\}$, $\{a4, b4\}$, then the underlined nodes are pruned from the tree in Figure 2.

4.4 Extracting the most general EPs

After pruning, only complete paths (from root to leaf) that are left in the tree are legal generalised EPs. However, these EPs do not necessarily represent the most general set since there might still exist some patterns that are specialisations of others. Since the paths in the tree have an imposed order as introduced previously, we can take out patterns from the tree in order from *right to left*, such that more general ones are always taken out first. In this way, when each EP is extracted it can be compared with those that were extracted earlier to see whether it is a specialisation of those.

For example in Figure 2, once $\{A2, B1\}$ has been extracted, $\{A2, b3\}$ is redundant, since $b3$ is a specialisation of $B1$. Finally, the set of generalised EPs obtained from each group of input leaf-level EPs are unioned together, in order to ensure that only the most general EPs are kept.

5 Evaluation

We now present experimental results to demonstrate the performance of our **GTree** algorithm on various data sets and with different taxonomy structures. The data sets used were acquired from the UCI Machine Learning Repository [1]. All experiments were performed on a 1GHz Pertium III machine, with 2GB of memory, running Solaris 86. The programs were coded in Java. Since none of the UCI data sets have taxonomy structures defined on their attribute values, we manually added three-level hierarchies for numerical attributes by aggregating values into increasing, non-overlapping ranges.

5.1 Dimensionality and scalability

Our goal is to evaluate the scalability of our algorithm by measuring its runtime performance on data sets with different dimensionalities. As a basis for comparison, we have implemented a **Brute-Force** algorithm, which enumerates all generalised patterns for each EP, and compares each generalised pattern against the negative class data. In terms of implementation, both algorithms used the **Border-Diff** algorithm for mining non-hierarchical EPs, although any other existing EP mining technique can be used. Therefore the total execution time recorded did *not* include the time taken for the mining of non-hierarchical EPs. The following table shows the experimental results of using the the two algorithms. Note that **GTree** is able to find all generalised EPs for all datasets, and in substantially less time than the brute force approach.

			Input		Output	Execution Times(s)	
Dataset	No. Attr	No. Hierarchical Attributes	No. EPs	No. Groups	No. G.EPs	Brute -Force	GTree
iris	4	4	9	4	6	0.16	**0.10**
glass	9	9	84	63	96	15.3	**1.0**
autos	25	15	662	658	675	108.6	**8.4**
breast-w	9	9	781	243	860	188.9	**7.1**
breast-cancer	9	3	949	302	933	7.4	**1.2**
diabetes	8	8	1015	224	1038	455.8	**12.0**
credit-a	15	6	3015	1375	2982	65.7	**13.8**
heart-statlog	13	6	4032	1792	4016	295.7	**16.8**
vehicle	18	18	34463	14253	36902	>10000	**994.1**
ionosphere	34	20	126330	79060	125434	>10000	**5035.2**

Table 1. Experimental results on 10 data sets

5.2 Compactness and support evaluation of generalised EPs

Having examined the efficiency of our algorithm, we now want to evaluate its effectiveness, in terms of the quantity and quality of the generalised EPs.

There are generally three classes of generalised EPs that occur, namely:

- **Class 1**: patterns consisting of leaf-level items only; e.g.$\{a1, b2\}$
- **Class 2**: patterns consisting of at least *one* non-leaf item and at least *one* leaf item; e.g. $\{a1, B2\}$, and
- **Class 3**: patterns consisting of non-leaf items only; e.g. $\{A1, B2\}$

Note that when we talk about non-leaf and leaf items, we refer to those attributes that are associated with hierarchies.

We now observe the quantity and quality of the patterns that belong to each of the three classes. The term *quality* refers to the expressiveness of each pattern. A pattern is considered powerful or expressive if it is sharply discriminative or strongly representative of the instances of a particular class, and this can be measured in terms of its support. The results shown in Table 2 are obtained from the same ten data sets which were used for producing Table 1. *Average support of each pattern* in the table reflects the average support of each pattern from a particular category, which is a direct indication of the expressiveness of a pattern. The results obtained are averaged over the ten data sets.

Property	Class 1	Class 2	**Class 3**
Proportion of the complete set	23.5%	51.4%	**25.1%**
Average support of each pattern	1.7%	3.9%	**10.7%**

Table 2. Experimental results showing the proportion of generalised EPs in each class

After post-processing, more than 76% of the output set are patterns that include items from higher levels in the taxonomy. Only less than 24% still remain

as leaf-level EPs, without being generalised. Table 2 shows that the average support of **Class 3** patterns is significantly higher than that of the patterns belonging to the other two classes. **Class-1** patterns, which are a subset of the original EPs, have the smallest average support. It is clear that **Class 3**, which corresponds to generalised EPs, has the highest average support. Overall, we find that the generalised EPs that are mined using our algorithm have much higher support than the traditional non-hierarchical EPs. This improvement means that the generalised EPs are representative of a larger proportion of the positive dataset, and are thus of higher quality than the original leaf-level EPs.

6 Conclusion and Future work

We have introduced the concept of *Generalised Emerging Patterns*. We have presented an algorithm for generating generalised EPs in data sets whose attributes are associated with membership hierarchies. Our experimental results based on ten real-life data sets show that our **GTree** algorithm is capable of mining generalised EPs from high-dimensional data sets within a small period of time. The complete set of most general generalised EPs mined from a data set has approximately the same size as its minimal non-hierarchical EP set. Moreover, the generalised EPs have higher support than traditional leaf-level EPs.

Acknowledgements. This research was supported in part by the Australian Research Council.

References

1. C. L. Blake and P. M. Murphy. UCI Machine Learning Repository.
2. H. Fan and K. Ramamohanarao. Efficient mining of emerging patterns: Discovering trends and differences. *PAKDD-2002*, 2002.
3. J. Li G. Dong and X. Zhang. Discovering jumping emerging patterns and experiments on real datasets. *IDC99*, 1999.
4. G.Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. *KDD-99*, pages 43–52, 1999.
5. R. Wirth J. Hipp, A. Myka and U. Guntzer. A new algorithm for faster mining of generalised association rules. *PKDD-98*, pages 74–82, 1998.
6. T.Manoukian J.Bailey and K.Ramamohanarao. Fast algorithms for mining emerging patterns. *PKDD-2002*, 2002.
7. J. Li. Mining emerging patterns to construct accurate and efficient classifiers. *PhD Thesis, The University of Melbourne*, 2001.
8. Y. Li and L. Sweeney. Learning robust rules from data. *Carnegie Mellon University, Computer Science Tech Report CMU ISRI 04-107, CMU-CALD-04-100*, Feb 2004.
9. T. Imielinski R. Agrawal and A. Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD-93*, pages 207–216, 1993.
10. R. Agrawal R.Srikant. Mining generalised association rules. *VLDB-95*, 1995.
11. G. Webb. Efficient search for association rules. *KDD-2000*, pages 99–107, 2000.
12. S. Zhang X. Wu, C. Zhang. Efficient mining of both positive and negative association rules. *ACM Trans. Inf. Syst. 22(3)*, pages 381–405, 2004.