Monitoring Patterns through an Integrated Management and Mining Tool

Evangelos E. Kotsifakos, Irene Ntoutsi, Yannis Vrahoritis, and Yannis Theodoridis

Department of Informatics, University of Piraeus,
80 Karaoli-Dimitriou St, GR-18534 Piraeus, Greece
{ek,ntoutsi,ytheod}@unipi.gr, jb@freemail.gr

Abstract. Patterns upon the data of many real applications are affected by changes in these data. We employ PATTERN-MINER tool to detect changes of clusterings extracted from dynamic data and thus, to provide insight on the dataset and to support strategic decisions. PATTERN-MINER, is an integrated environment for pattern (data mining model) management and mining that deals with the whole lifecycle of patterns, from their generation (using data mining techniques) to their storage and querying, putting also emphasis on the comparison between patterns and meta-mining operations over the extracted patterns. In the current version, PATTERN-MINER integrates also an algorithm and technique for monitoring patterns (currently clusters) over time.

Keywords: Pattern management, Pattern-base, pattern comparison, pattern monitoring.

1 Introduction

Clustering techniques are used to find interesting groups of similar objects in large datasets. As the data in most databases are changing dynamically, clusters upon these data are affected. A lot of research has been devoted in adapting the clusters to the changed dataset. On the other hand research has expanded and it is devoted in tracing of the cluster changes themselves, trying thus to reveal knowledge about the undelying dataset to support strategic decisions. For example, if a business analyst who studies customer profiles, could understand how such profiles change over time he/she could act towards a long-term proactive portfolio design instead of reactive portfolio adaptation.

We demonstrate Pattern-Miner, an integrated environment that deals with pattern modeling, storage and retrieval issues using state-of-the-art approaches in contrast to existing tools that deal with specific aspects of the pattern management problem, mostly storage. Pattern-Miner offers an environment that provides the capability not only to generate and manage the different types of patterns in a unified way, but also to apply more advanced operations over patterns, such as comparison, meta-mining and cluster monitoring without facing interoperability or incompatibility issues as if using different applications for each task. Pattern-Miner follows a modular architecture and integrates the different Data Mining components offering transparency to the end user. A previous version of Pattern-Miner has been demonstrated at KDD2008 conference. In the current version, a major addition has been made. A module for monitoring patterns over time. At first we are dealing with cluster monitoring as it has wide application to many scientific or commercial fields.

In order to better understand the theoretical background of PATTERN-MINER we briefly present some basic notions on patterns, following the PBMS approach [5]. A *pattern* is a compact and rich in semantics representation of raw data. Patterns are stored in a *pattern base* for further analysis. The pattern base model consists of three layers: pattern types, patterns, and pattern classes. A *pattern type* is a description of the pattern structure, e.g. decision trees, association rules, etc. A pattern type is a quintuple $pt = (n, ss, ds, ms, f)$, where n is the name of the pattern type, ss (structure schema) describes the structure of the pattern type (e.g. the head and the body of an association rule), ds (source schema) describes the dataset from which patterns are extracted, ms (measure schema) defines the quality of the source data representation achieved by patterns (e.g. the support and the confidence in case of an association rule pattern) and f is the formula that describes the relationship between the source data space and the pattern space. A *pattern* is an instance of the corresponding pattern type and a *class* is a collection of semantically related patterns of the same type.

2 Extended Pattern-Miner Architecture

Figure 1 depicts the PATTERN-MINER architecture, including the pattern-monitoring module. *PATTERN-MINER engine* lies in the core of the system arranging the communication between the different peripheral components providing also the end user interface.

In this section, we provide an overview of the funcionality of each module of PATTERN-MINER and we focus on the *Pattern monitoring module* that consists the most recent and advanced module.

Pattern extraction and representation: The *Data Mining engine* component is responsible for the extraction of patterns. We employ for this task *WEKA*, since it is an open source tool and offers a variety of algorithms for different Data Mining tasks

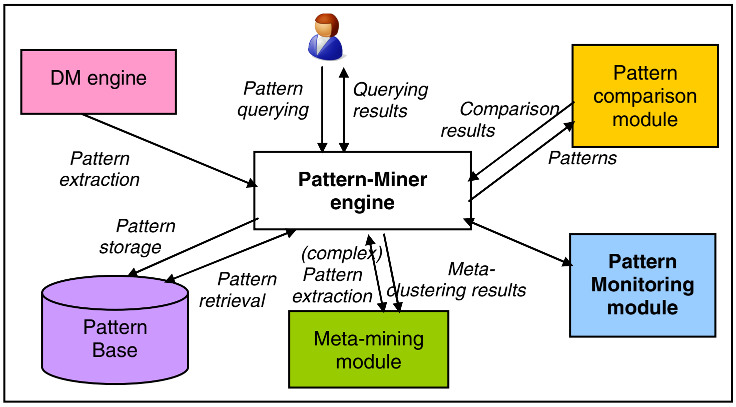


Fig. 1. The PATTERN-MINER architecture

as well as preprocessing capabilities over raw data. The output of the Data Mining process is represented with respect to the PBMS approach, described above. Several schemes have been proposed in the literature for the representation of patterns. The most popular choice is *PMML* [3], an XML-based standard that allows the definition of Data Mining and statistical models using a vendor-independent method. Different models are described through different XML schemes. In PATTERN-MINER, we adopt PMML for pattern representation and, thus, we convert the output of the *Data Mining engine* component into PMML format.

Pattern storage and querying: Since patterns are represented as XML documents (through PMML), a native XML database system is used for their storage in the *Pattern Base*. In particular, we employ the open source *Berkeley DBXML*, which comprises an extension of the Berkeley DB with the addition of an XML parser, XML indexes and the XQuery data query language. PATTERN-MINER provides a basic environment for querying the pattern base, through the XQuery language. Regarding the supported query types, the user can retrieve the whole pattern or some component of the pattern (measure and/ or structure), as well as to impose constraints over these components. The results are displayed in his/her screen and can be stored in the file system for future analysis.

Pattern comparison: One of the most important operations on patterns is that of *pattern comparison* with applications like querying (e.g. k-nearest neighbor queries) and change detection upon dynamic data [4]. Recognizing this fact, we distinguish the comparison process from the querying process and we implement it separately through the *Pattern comparison module*. The comparison is carried out on the basis of *PANDA* [1], a generic and flexible framework for the comparison of patterns defined over raw data and over other patterns as well. Comparison utilizes both structure and measure components of patterns. The user defines the patterns as well as the way that they should be compared, i.e. how the different components of PANDA are instantiated. The output is a dissimilarity score accompanied with a justification, a report actually of how the component patterns have been matched.

Meta-mining: Due to the large amount of extracted patterns, several approaches have lately emerged that apply Data Mining techniques over patterns instead of raw data, in order to extract more compact information. The *Meta-mining module* takes as input a set of different clustering results extracted from the same dataset (through different clustering algorithms or different parameters) or from different datasets (through from the same generative distribution) and applies Data Mining techniques over them, in order to extract *meta-patterns*. So far, the meta-mining component focuses on meta-clustering [2], i.e. grouping of clustering results into groups of similar clusterings. The user has full control of the clustering process by choosing the similarity function and the clustering algorithm. The extracted meta-patterns can be stored in the pattern base for further exploitation.

Pattern Monitoring: While PATTERN-MINER is a tool for managing all types of patterns, at the current moment we have implemented a Cluster Monitoring technique that is based on the theory and algorithm described in [4]. In this approach, the transitions of clusters extracted upon an accumulating dataset are traced and modeled. Clustering occurs at specific timepoints and a “data ageing” function can be used to assigns lower weights to all or some of the past records. The set of features used for clustering may also change during the period of observation, thus allowing for the inclusion of new features and the removal of obsolete ones. PATTERN-MINER assumes re-clustering rather than cluster adaptation at each timepoint, so that both changes in existing clusters and new clusters can be monitored. Transitions can be detected even when the underlying feature space changes, i.e. when cluster adaptation is not possible. Terms like cluster match, cluster overlap, cluster transition and lifetime of a cluster are core notions of cluster monitoring. This module exploits the clusterings that are stored in the pattern-base and employs the query and comparison capabilities of the system.

3 Demo Description

PATTERN-MINER is a tool that can be used in a lot of different areas, scientific or commercial. To point out the major advantages of the integrated environment of PATTERN-MINER we demonstrate a simple senario of a supermarket and its manager as the end-user. The supermarket has a database and everyday transactions are stored in it. The manager is interested in finding useful patterns in the data, like associations in the purchase of the products and clusters of customers with specific profiles and buying habits. Except from these simple patterns, the manager is interested in comparing clusters of customers or products discovered from the same dataset. Moreover, the manager wishes to monitor clusters of customer profiles over time, so he/she can capture any changes in buying preferences or habits. Are there any new clusters that describe a different customer profile? Some clusters may have been disappeared or shrinked while others could have been merged or expanded. PATTERN-MINER can be used to answer these questions supporting the manager on important decisions about strategies, campaigns, supplies etc. In the following paragraphs we describe the steps that the manager as the end-user would follow, to process the data from the dataset of the supermarket in order to extract and manage interesting patterns.

Pattern extraction and storage: The manager defines the data source (supermarket database), the Data Mining algorithm and its parameters. He/she would choose the apriori algorithm to find for example associations between products. To find clusters over the customer demographics to create profiles, the K-Means or the EM clustering algorithm would be appropriate. The extraction takes place in WEKA and the results are converted into PMML format before being stored in a user-specified container in the XML pattern base as well as, in a file on the hard disk.

Pattern query: The user defines, in the "Query pattetrn base" tab, the pattern set to be queried and the query in the Xquery language. PATTERN-MINER *engine* creates the connection to the pattern base, executes the query and returns the results to the user and also saves them to a file. A sample query is shown in **Figure 2**, described in both natural language and Xquery.

Pattern comparison: This module allows the user to define the patterns to be compared, e.g. sets of rules extracted from different months or clusters describing customer profiles. Then, the user chooses the appropriate comparison function from the candidate functions implemented for each pattern type. The results are returned to the user, who can detect any changes in the sales-patterns and decide whether these changes were expected (based on company's strategy) or not (indicating some suspicious or non-predictable behavior).

Query (natural language):

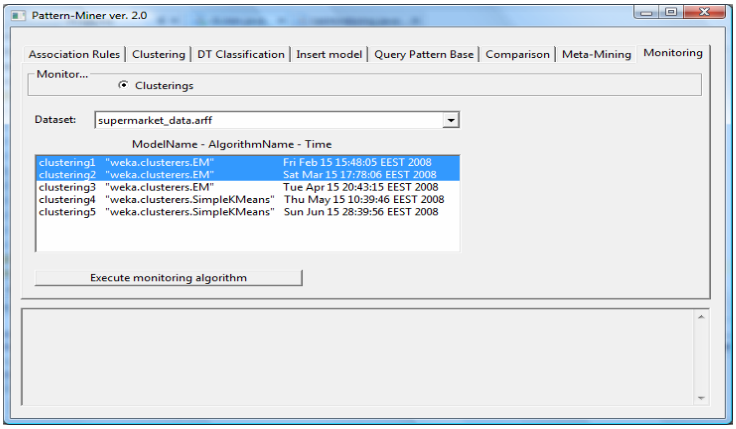
Retrieve the clusters from the super_market dataset that have been extracted using EM algorithm.

Query (XQuery):

```
declare namespace a="http://www.dmg.org/PMML-3_2";
collec-
tion("Clustering.dbxml")[dbxml:metadata("dbxml:dataFileName")="C:\Patter
nMiner\data_files\super_market_data.ARFF"]/a:PMML/a:ClusteringModel[@alg
orithmName="weka.clusterers.EM"]
```

Fig. 2. A sample query for the Clustering pattern-model

Meta-mining: The user defines the pattern sets to be used as input to the *Meta-mining module* (e.g. sets of rules extracted at each month of 2007), selects the clustering algorithm/ parameters, as well as the similarity measure between sets of rules. The input sets are clustered into groups of similar sets of rules (e.g. March and April are placed in the same group, since they depict similar buying behavior), which can be also stored in the pattern base for future use. The manager can exploit these results in order to decide similar strategies for months belonging to the same cluster.



Monitor... [✓ Clusterings]

Dataset: supermarket_data.arff

ModelName - AlgorithmName - Time

```
clustering1  "weka.clusterers.EM"            Fri Feb 15 15:48:05 EEST 2008
clustering2  "weka.clusterers.EM"            Sat Mar 15 17:78:06 EEST 2008
clustering3  "weka.clusterers.EM"            Tue Apr 15 20:43:15 EEST 2008
clustering4  "weka.clusterers.SimpleKMeans"  Thu May 15 10:39:46 EEST 2008
clustering5  "weka.clusterers.SimpleKMeans"  Sun Jun 15 28:39:56 EEST 2008
```

Fig. 3. Cluster Monitoring screenshot

Cluster Monitoring: User defines the dataset from which the clusters have been extracted. A list of all the clusterings that have been carried out over the spesific dataset is available to the user, sorted by the extraction time. The supermaket manager wants to observe the customer profiles over time. Choosing the apropriate dataset (supermarket.arff), Pattern-Miner returns all the different clusterings that have been created from that dataset, along with the clustering algorithm and the extraction time. The manager

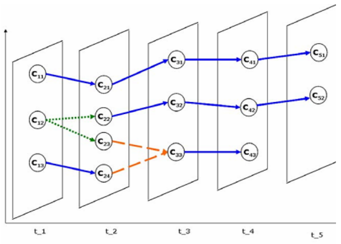


Fig. 4. Graphical reprepsentation of cluster monitoring output

chooses two or more clusterings and runs the cluster monitring process. This process results in a matrix showing the clusters of the first clustering and their changes over time (new clusters, clusters that no longer exists, shrinked or expanded clusters etc). Currently the output is in text format, representing the graph depicted in Figure 4.

4 Conclusions and Outlook

Advanced operations over patterns, like comparison, meta-mining and cluster monitoring while important for users of a variety of fields, are not supported from data mining or database systems. PATTERN-MINER is an integrated environment for pattern management that supports the whole lifecycle of patterns and also offers sophisticated comparison, meta-mining and cluster monitoring operations over patterns. It follows a modular architecture that employs state-of-the-art approaches at each component. Its advantage lies in the fact that all the operations related to the management of patterns (as data mining results) are integrated into one system in transparent to the user way. It is open source and easily expandable while, because of the use of PMML files, the exhange of data and results with other systems is a simple issue.

Several improvements though, can be carried out: First, existing components can be enhanced, e.g. querying could be improved through appropriate indices and new query types could be supported. Also, the *Meta-mining* and *cluster monitoring modules* can be extended so as to support more pattern types, like decision trees. Second, new components can be added, e.g. some visualization module for better interpretation of the results. Except for the scenario we described, other potential applications include cluster-based image retrieval, pattern validation, comparison of patterns extracted from different sites in a distributed environment setting, etc.

References

1. Bartolini, I., Ciaccia, P., Ntoutsi, I., Patella, M., Theodoridis, Y.: A Unified and Flexible Framework for Comparing Simple and Complex Patterns. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 496–499. Springer, Heidelberg (2004)
2. Caruana, R., Elhawary, M., Nguyen, N., Smith, C.: Meta Clustering. In: Proc. ICDM (2006)
3. DMG - PMML, `http://www.dmg.org/pmml-v3-1.html`
4. Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., Schult, R.: MONIC: Modelling and monitoring cluster transitions. In: KDD (2006)
5. Terrovitis, P., Skiadopoulos, S., Bertino, E., Catania, B., Maddalena, A., Rizzi, S.: Modeling and language support for the management of pattern-bases. Data Knowl. Eng. 62(2) (August 2007)