IEEE 6th Int. Conf. on Data Mining (ICDM'06), Hong Kong, China, 2006, pp.917-921

COSMIC: Conceptually Specified Multi-Instance Clusters

Hans-Peter Kriegel Alexey Pryakhin Matthias Schubert Arthur Zimek
Institute for Informatics, Ludwig-Maximilians-Universität München, Germany
http://www.dbs.ifi.lmu.de
{*kriegel,pryakhin,schubert,zimek* }@*dbs.ifi.lmu.de*

Abstract

Recently, more and more applications represent data objects as sets of feature vectors or multi-instance objects. In this paper, we propose COSMIC, a method for deriving concept lattices from multi-instance data based on hierarchical density-based clustering. The found concepts correspond to groups or clusters of multi-instance objects having similar instances in common. We demonstrate that COSMIC outperforms compared methods with respect to efficiency and cluster quality and is capable to extract interesting patterns in multi-instance data sets.

1. Introduction

More and more data mining applications employ sets of feature vectors or multi-instance (MI) objects to represent a single data object. For example, a molecule can be described by the set of all conformations or shapes it might adopt, or a web site can be described by a set of webpages. In general, an MI object is represented by a set of object descriptions of one and the same type, e.g. color histograms or text vectors. We will call the elements of an MI object *instances*. In this paper, we propose COSMIC, a method for deriving **CO**nceptually **S**pecified **M**ulti-**I**nstance **C**lusters. The idea of COSMIC is to derive a so-called concept lattice as known from formal concept analyzes [4] to describe the rich relationships between sets of feature vectors. Based on the concept lattice we can derive flat as well as hierarchical clusterings.

In formal concept analysis, each object o can be described by a set of nominal attributes $Desc(o)$. A concept C is defined by a set of objects and a set of attributes $Desc(C)$ if for each object $o \in C$, $Desc(C) \subseteq Desc(o)$ holds. In other words, a concept is the maximal set of objects that can be described by $Desc(C)$. Additionally, concepts can be specialized and generalized to sub- or super concepts by adding or dropping attributes which decreases or increases the set of objects that are covered by the concept. Therefore, the set of all concepts for a given set of objects and attributes is organized in the so-called concept lattice.

To derive a concept lattice from a set of MI objects, each object has to be described by a set of nominal attributes drawn from the corresponding instances. Thus, we have to find groups of instances having a similar meaning, i.e. we cluster the instances. Each cluster of instances does now provide a so-called *concept attribute* (CA) and an MI object can be described by the set of CAs its instances belong to. The clustering algorithm used for deriving CAs should only group objects into a cluster that are really similar and should not require to specify the number of clusters to be found in the data set. Thus, partitioning clustering algorithms are not suitable for the given task. Therefore, COSMIC relies on hierarchical density-based clustering [1] which is capable to find an arbitrary number of clusters and distinguishes noise instances. Furthermore, the found cluster hierarchy describes the relationship between the found CAs. After clustering the instances, COSMIC extracts the CAs that are useful for describing formal concepts from the resulting reachability plot and lists all concepts that can be found in the given data set. The resulting concept lattice can now be used to derive flat and hierarchical clusterings of the given MI data set.

The rest of the paper is organized as follows. In Section 2, we will survey previous work in data mining with MI objects. Section 3 provides the necessary formal framework for our approach to clustering MI objects. Afterwards, Section 4 describes the complete COSMIC algorithm for generating a concept lattice. Section 5 displays the results of our experimental evaluation, and Section 6 concludes the paper with a summary.

2. Related Work

Data mining in multi-instance objects has so far been predominantly studied w.r.t. classification (for a survey cf. [12]). For clustering, MI objects were handled by complex distance measures [3, 9] or kernel functions [5]. Employing these similarity measures, it is possible to employ distance-based data mining approaches like k-NN classification, k-medoid Clustering [6] or OPTICS [1], or kernel methods [5]. Clustering data objects based on a concept lattice was previously used for other data types such as text data [10]. However, to the best of our knowledge none of these methods deals with MI objects and the question of how to derive CAs from a set of MI objects. In [7], a method to derive MI clusters based on EM clustering was proposed. The method clusters the instances using ordinary EM clustering and afterwards uses a multinomial process to group MI objects. Though this method displayed promising results, finding a meaningful clustering strongly depends on the input parameters. COSMIC employs a modification of the hierarchical density-based clustering algorithm OPTICS [1] to generate a reachability plot and to derive a cluster hierarchy.

3. Preliminaries

Let F be a feature space. Then, $i \in F$ is called an *instance* in F. A *multi-instance (MI) object* o is given by an arbitrary sized set of instances $o = \{i_1, \ldots, i_k\}$ where $i_j \in F$. To denote the unique MI object an instance i belongs to, we will write *MiObj(i)*. For example, a web site is an MI object and its instances are the web pages within this site.

To derive a concept lattice, we need to transform MI objects into objects that are described by a set of nominal attributes. Thus, we employ clustering to group several instances to so-called *concept attributes (CAs)*. A CA c describes a set of similar instances $\mathcal{I}_c \subset F$. For any $i \in \mathcal{I}_c$, we will denote $\mathit{ConAttr}(i) = c$.

Each CA c can now be considered as a nominal attribute describing each MI object containing at least one element of $\mathcal{I}_c$. As mentioned above, we consider the CAs to be organized in a hierarchy. Thus, a CA might generalize several more specialized CAs. Consider for example the CA "product descriptions". Subconcept Attributes (SubCAs) might be "descriptions of hardware products" and "descriptions of software products". Let s, c be two concept attributes in F where $\mathcal{I}_s \subseteq F$ and $\mathcal{I}_c \subseteq F$ are the sets of members of s and c, respectively. Then, s is called *subconcept attribute* (SubCA) of c, denoted by $\mathit{SubCA}_c(s)$ iff $\mathcal{I}_s \subset \mathcal{I}_c$. Additionally, s is called *direct subconcept attribute* of c iff $\nexists r : \mathit{SubCA}_c(r) \wedge \mathit{SubCA}_r(s)$.

To define a hierarchy, we start with one root CA c_{all} containing all instances in F. Then, all CAs except for the root CA are a SubCA of at least one other CA. Let $H = \{c_1, ..., c_n\}$ be a set of concept attributes in F. H is called a concept hierarchy if $F = c_{all} \in H$.

Having derived a mapping of instances to CAs, we now will formalize the resulting concept lattice as introduced in formal concept analysis. Therefore, we will first of all introduce a formal context (similar to [4]). Let $\mathcal{D}$ be a set of objects and let H be a CA hierarchy. A formal hierarchical context is now given by the triple $(\mathcal{D}, H, I)$ where I is a binary relation between $\mathcal{D}$ and H: $I \subseteq (\mathcal{D} \times H)$, and the following condition holds:
$\forall c_i, c_j \in H, \forall o \in \mathcal{D}:$
$\mathit{SubCA}_{c_j}(c_i) \wedge (o, c_i) \in I \Rightarrow (o, c_j) \in I$.

Thus, the context defines which CAs are contained in which object. Furthermore, the condition states that if an object is described by a CA c_i, then it must also be described by all of the ancestors of c_i in the CA hierarchy.

To describe the output of COSMIC, we first of all need to specify a single concept. Let $(\mathcal{D}, H, I)$ be a formal hierarchical context. An object set $C \subseteq \mathcal{D}$ together with a CA set $\mathit{Desc}(C) \subseteq H$ is called concept if the following conditions hold:

$$\begin{array}{ll} (1) & C = \{o \in \mathcal{D} | \forall a \in \mathit{Desc}(C) : (o, a) \in I\} \\ (2) & \mathit{Desc}(C) = \{a \in H | \forall o \in C : (o, a) \in I\} \end{array}$$

We will call $\mathit{Desc}(C)$ the concept description of C.

In other words, a concept is the maximal subset of the objects which contains all elements of the concept description $\mathit{Desc}(C)$. For example, a concept of websites could be described by the concept attributes "employment", "financial reports", and "software development". Each website belonging to the concept must contain at least one web page belonging to each of these CAs. At the same time there is no website in the given context being described by these concept attributes that is not part of the concept. Since a concept that is not general enough is not useful for examining patterns in a data set, we will call the cardinality of a concept C the support of C, denoted by *support*(C). For building a concept lattice, it is therefore often sufficient to only consider concepts that have a support above a certain minimum threshold *MinSup*.

We will now specify subconcepts. Let C_1, C_2 be two concepts in $\mathcal{D}$ w.r.t. to the context $(\mathcal{D}, H, I)$. Then C_1 is called *subconcept* of C_2, denoted by $\mathit{SubConcept}_{C_2}(C_1)$ iff $C_1 \subset C_2 \Leftrightarrow \mathit{Desc}(C_1) \supset \mathit{Desc}(C_2)$

For example, a concept of web sites which is described by the CAs "faculty pages" and "lectures" is a generalization of the subconcept being described by "faculty pages", "lectures" and the additional CA "computer science lectures".

Finally, we can specify the concept lattice for a given context. Let $(\mathcal{D}, H, I)$ be a formal context. The set of all concepts that can be found in the context $(\mathcal{D}, H, I)$ together with the subconcept relation between these concepts is called concept lattice.

The resulting concept lattice now describes an overlapping hierarchical grouping of an MI data set that can be explored directly. Additionally, we can use the concept lattice to derive a flat disjunctive clustering by assigning each MI object to the most specialized concept it belongs to. The goal of COSMIC is to derive a concept lattice over a data set of MI objects containing all concepts having at least a support of *MinSup*. The CAs this lattice is based on are organized in a hierarchy and it is guaranteed that each CA in this hierarchy is employed to describe at least one cluster. While processing, COSMIC avoids considering useless candidates for CAs whenever possible. Hence, COSMIC is rather efficient.

4. COSMIC

The input of COSMIC is a set of MI objects over an arbitrary feature space F. Additionally, we need a distance measure $dist : F \times F \rightarrow \mathbb{R}^+$ for comparing the instances. The result of COSMIC is a concept lattice which is defined on the basis of a CA hierarchy. COSMIC proceeds in two steps. The first step is based on the density-based notion of clustering and, thus, needs the parameters which are specific for this approach, i.e. μ and ε. In the second step, we only need to specify the minimum support of the concepts called *MinSup*.

Deriving a Concept Hierarchy In the first step of COSMIC, the set of all instances $\mathcal{I} = \bigcup_{o \in \mathcal{D}} o$ is clustered with the density-based clustering algorithm OPTICS [1]. However, to avoid CAs describing a single or only a small number of MI objects, we modified the definition of the core-distance in order to find CAs which are dependent on at least μ MI objects instead of μ arbitrary instances.

Definition 1 (Concept core-distance)
Let $\mu \in \mathbb{N}$, $\varepsilon \in \mathbb{R}^+$ and let $\mathcal{D}$ be a set of MI objects and $\mathcal{I} = \bigcup_{o \in \mathcal{D}} o$. The μ-nearest MI neighbors of an instance i are the smallest set $N_\mu^{MI}(i) \subseteq \mathcal{I}$ that contains (at least) μ instances for which the following conditions hold:

$$\begin{aligned} (1) \quad & \forall p \in N_\mu^{MI}(i), \forall q \in \mathcal{D} \setminus N_\mu^{MI}(i) : \\ & dist(p,i) < dist(q,i) \\ (2) \quad & |\{MiObj(x) | x \in N_\mu^{MI}(o)\}| \geq \mu \end{aligned}$$

Then, $dist_\mu(i) = \max\{dist(i,q) \mid q \in N_\mu^{MI}(i)\}$, and the concept core-distance *of instance i, denoted by $ConceptCoreDist_\mu^\varepsilon(i)$, is defined as follows:*

$$ConceptCoreDist_\mu^\varepsilon(i) = \begin{cases} dist_\mu(i) & : \; dist_\mu(i) \leq \varepsilon \\ \infty & : \; dist_\mu(i) > \varepsilon \end{cases} .$$

The resulting reachability plot implies a hierarchical clustering of instances. However, not all of the clusters might be useful for describing a concept and are thus suitable CAs. Therefore, COSMIC collects so-called hot spots while generating the plot itself. The hot spots mark positions in the plot where a more general instance cluster can be separated into more specialized clusters. The hot spots are used in the next step to specialize already found CAs and thus to derive new subconcepts. Technically, a hot spot is a position in the reachability plot, where the reachability distance is smaller on the right side and is smaller or equal on the left side. Thus, a hot spot corresponds to a peak or the rightmost position of a plateau in the reachability plot. Two examples of hot spots are illustrated in Figure 1.

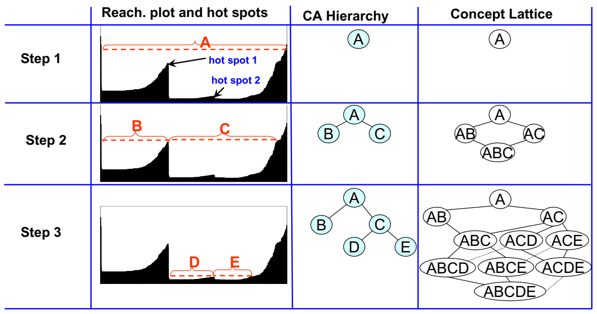

Figure 1. Example of derived CA hierarchy and concept lattice.

Deriving Attributes and Concepts The second step of COSMIC generates all concepts that contain at least *MinSup* MI objects. *MinSup* is the only parameter that has to be specified for the actual extraction of MI clusters.The input for the second part of COSMIC is the reachability plot derived in the first step and the hot spots collected within this plot. The general idea of this algorithm is to employ a top-down sweep line algorithm to the reachability plot which simultaneously extracts CAs and concepts. The algorithm starts with a trivial set of one CA, namely c_{all}, and a trivial concept which corresponds to the complete dataset $\mathcal{D}$ and is described by $\{c_{all}\}$. Before starting the sweep line algorithm, we first of all sort the hot spots descending w.r.t. their reachability distance in the plot. The sweep line stops at each hot spot in this generated order and determines the CA the hot spot is contained in. In the following, we will refer to this CA as the split CA. If the split CA is not used for describing any concept, we already can examine the next hot spot because there cannot be any concept being described by any SubCA of the split CA. This observation can be formalized in the following monotonicity criterion: Let $C = \{c_1, \ldots, c_m\}$ be a set of CAs over the feature space F and $M = \{M_1, \ldots, M_l\}$ be a set of concepts which are described by C. Furthermore, let $c_g \in C$ be some CA and let $M_g \in M$ be some concept with $c_g \in Desc(M_g)$. Then for any subconcept M_s that can be described by

$$Desc(M_g) \setminus c_g \cup SubCA(c_g)$$

the following rule holds: $|M_G| \leq k \Rightarrow |M_s| \leq k$.

If the split CA is an element of any concept description, the algorithm determines the expansion of the new SubCAs. Therefore, the plot is traversed in both directions beginning with the hot spot and ending with a position for which the reachability distance is again at least as large as at the hot spot. Let us note that it might be necessary to cross some plateau, i.e. an area of the plot having the same reachability distance, before finding the indicated cluster. The CAs are now stored in the CA hierarchy below the split CA. Since it is possible to split a CA which is an inner node of the CA hierarchy, the degree of the CA hierarchy is arbitrary.

After determining the SubCAs of the split CA, COSMIC has to check if it is possible to extend any concept that is described by the split CA. Thus, all concepts being described by the split CA are checked if they can be specialized into subconcepts that can be described by any of the new SubCAs or the combination of both. If any of the resulting cluster descriptions denotes a cluster having more than *MinSup* elements, the concept lattice is extended. In the first two cases, the new subconcept can be described by one of the SubCAs and the new concepts are direct subconcepts of the concept that is currently examined. In the third case, the existence of a concept containing both SubCAs implies that both previous cases form also a subconcept because the new concept contains the intersection of the subconcepts generated in the previous cases. Thus, if the concept being described by both SubCAs has more than *MinSup* objects, then the concepts containing only one of the SubCAs must also have at least *MinSup* members. Let us note that determining the cardinality of the subconcepts can be done quite efficiently. For every element of a concept that might be split, we simply have to check if it has at least one instance in any of the new subconcepts and then combine the results. After a concept is processed that contains the split CA in its description, additional links have to be added to the concept lattice between each pair of newly constructed concepts for which there was a subconcept relation between their father concepts.

The algorithm terminates when there are no more hot spots that could be processed. Figure 1 illustrates the process of CA extraction and concept expansion on a simple example having a plot based on two hot spots. The left column displays hot spots and corresponding concept attributes in the reachability plot, the middle column displays the CA hierarchy that can be derived from this plot. The right column contains all possible concept descriptions that can be derived from the CA hierarchy.

5. Evaluation

All experiments were carried out on a workstation equipped with 2 Opteron 1.8 GHz processors and 8 GB memory. The compared algorithms are implemented in Java 1.5.

Table 1 summarizes the 5 real-world data sets employed in the Experiments. The MUSK 1 and MUSK 2 data sets were taken from the UCI repository [8] and describe a set of molecules. The Dobson&Doig (DS 3) and BRENDA (DS 4) data sets consist of enzyme data from the protein data bank (PDB)[1]. For each subunit in a protein, we derived a histogram over the occurrences of the 20 amino acids and additional 6 exchange groups. The class labels for DS 3 were obtained as described in [2] and the labels for DS 4 correspond to 3rd level enzyme class numbers of BRENDA[2]. The last data set (DS 5) consists of health care web sites taken from WebKB[3]. Each page was described by 8,000 dimensional feature vector.

We compared COSMIC to density-based and k-medoid clustering algorithms working on set-valued distance functions. Therefore, we compared the effectiveness and the efficiency of COSMIC with that of PAM and OPTICS. To enable PAM and OPTICS to compare MI objects, we used the Hausdorff distance (HD) [3], the minimum Hausdorff distance (mHD) [11] and the Sum of Minimum Distances (SMD) [3].

In order to find a unique mapping of MI objects to one dedicated cluster, we determine the most specialized concept, i.e. the concept having the most CAs in its description. If this method does not provide a unique mapping, we additionally weight each CA with the inverse number of instances supporting the CA and calculate the sum over all CAs in the concept description. To assess the quality of these clusters, we determined the majority class in each cluster and calculated the precision. Finally, we computed the weighted sum using the cluster sizes as weights.

To compare COSMIC to OPTICS, we calculated the cluster hierarchy of OPTICS by applying a cluster extraction method that is based on hot spots. On the resulting clustering, we mapped each object to its most specialized cluster and summed up the precision of the most special cluster. We compared all results of the compared clustering approaches w.r.t. an equal number of clusters. On the average, the derived number of clusters was about four times the number of classes.

For all data sets, COSMIC achieved a higher precision than the other methods (cf. Figure 2). This suggests that the MI clusters derived from the concept lattice are more precise than using established set-valued distance functions. For example, for DS 1 COSMIC achieved a precision of 0.858, whereas the best result of the remaining methods was 0.772.

[1]http://www.rcsb.org/pdb/

[2]http://www.brenda.uni-koeln.de/

[3]http://www.cs.cmu.edu/afs/cs.cmu.edu/project/theo-11/www/wwkb/

Table 1. Description of the test data sets.

	Data Set 1 (DS 1)	Data Set 2 (DS 2)	Data Set 3 (DS 3)	Data Set 4 (DS 4)	Data Set 5 (DS 5)
Name	MUSK 1	MUSK 2	Dobson & Doig	Brenda	Web
# MI-Obj.	92	102	969	10,254	46
# Inst. / MI-Obj.	5.2	64.7	2.4	1.977	7.72
# Classes	2	2	7	115	4

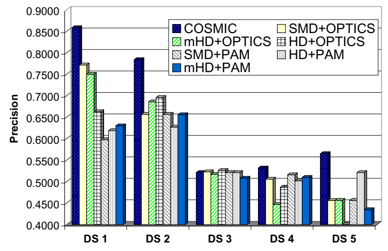

Figure 2. Average Cluster Precision.

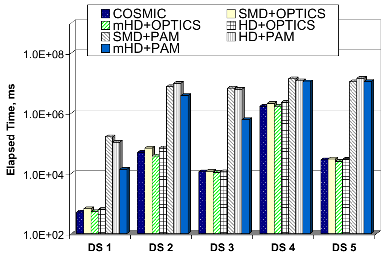

Figure 3. Complete Runtime of COSMIC and its comparison Partners.

In the following, we will describe an example concept found in the concept lattice of the website data set DS 5. The cluster hierarchy displayed several leaf concepts like *Menu Pages*, *Contact Pages*, *Employment Pages*, *Quarter Results*, *Disclaimers* and *Company Descriptions*. We identified a concept of websites that were described by *Employment Pages*, *Company Descriptions* and *Contact Pages*. The member websites of this concept represent companies from the biotech area that were trying to recruit new employees.

To measure the efficiency of COSMIC and its competitors, we compared the elapsed runtime (cf. Figure 3). COSMIC showed a runtime behavior which is comparable to the best of the remaining methods. Another interesting result can be observed when comparing the runtimes of the two steps of COSMIC. The time that was spent on deriving the reachability plot from the set of all instances took on the average about two orders of magnitude more time than the second step deriving the concept lattice from the plot. Thus, running COSMIC only requires an marginal additional amount of time compared to running OPTICS on the set of all instances.

6 Conclusion

In this paper, we proposed COSMIC, a method for deriving concept lattices from MI data sets. COSMIC describes concepts of MI objects by sets of so-called cluster attributes (CAs). A CA is a common pattern in the data space of instances that might be used to characterize at least μ MI objects. A CA hierarchy is calculated employing density-based hierarchical clustering. The second step of COSMIC extracts a concept lattice along with the CA hierarchy used for the concept descriptions. The results of our experiments demonstrate that COSMIC generates more precise clusterings w.r.t. a reference class set and that COSMIC scales well to even larger data sets.

References

[1] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *Proc. SIGMOD*, 1999.

[2] P. D. Dobson and A. J. Doig. Predicting enzyme class from protein structure without alignments. *J. Mol. Biol.*, 4(345):187–199, 2005.

[3] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):103–133, 1997.

[4] B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1999.

[5] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *Proc. ICML*, pages 179–186, 2002.

[6] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Academic Press, 2001.

[7] H.-P. Kriegel, A. Pryakhin, and M. Schubert. An EM-approach for clustering multi-instance objects. In *Proc. PAKDD*, 2006.

[8] D. Newman, S. Hettich, C. Blake, and C. Merz. UCI repository of machine learning databases, 1998.

[9] J. Ramon and M. Bruynooghe. A polynomial time computable metric between points sets. *Acta Informatica*, 37:765–780, 2001.

[10] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proc. SIGIR*, pages 206–213, 1999.

[11] J. Wang and J. Zucker. Solving multiple-instance problem: A lazy learning approach. In *Proc. ICML*, pages 1119–1125, 2000.

[12] Z.-H. Zhou. Multi-instance learning: A survey. Technical report, AI Lab, Computer Science and Technology Department, Nanjing University, Nanjing, China, 2004.