Conceptual Clustering of Multi-Relational Data

Nuno A. Fonseca[1], Vitor Santos Costa[1,3], and Rui Camacho[2]

[1] CRACS-INESC Porto LA, Universidade do Porto,
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal
[2] LIAAD-INESC Porto LA & DEI-FEUP, Universidade do Porto,
Rua Dr Roberto Frias s/n, 4200-465 Porto, Portugal
[3] DCC-FCUP, Universidade do Porto,
Rua do Campo Alegre 1021/1055, 4169-007 Porto, Portugal

Abstract. "Traditional" clustering, in broad sense, aims at organizing objects into groups (clusters) whose members are "similar" among them and are "dissimilar" to objects belonging to other groups. In contrast, in conceptual clustering the underlying structure of the data together with the description language which is available to the learner is what drives cluster formation, thus providing intelligible descriptions of the clusters, facilitating their interpretation.
We present a novel conceptual clustering system for multi-relational data, based on the popular $k - medoids$ algorithm. Although clustering is, generally, not straightforward to evaluate, experimental results on several applications show promising results. Clusters generated without class information agree very well with the true class labels of cluster's members. Moreover, it was possible to obtain intelligible descriptions of the clusters.

1 Conceptual Relational Clustering

We start by defining the learning task here addressed.
Given:

- a (finite) set of examples (E), with each example encoded as a ground atom;
- background knowledge (B) encoded as statements in Prolog;
- concept description language ($\mathcal{L}$) specifying argument types, modes, and arity of all predicates in E and B together with constraints on the clauses that can be derived from B and E (e.g.,. maximum number of literals);
- a *distance function* ($d(e_i, e_j)$) that computes the similarity between any two examples $e_i,\ e_j\ \in E$;
- optionally, a *number of clusters* k ($k <\mid E \mid$);

Find:

- a partition C (set of clusters) on the examples;
- that maximizes a given *quality criteria*;

Usually, clustering is performed by mapping examples into points in an n-dimensional space and then using a measure such as euclidian distance. Within the context of ILP, several propositionalisation methods may be used toward this purpose. On the other

hand, quite often multi-relational data needs to be mapped into a very highly-dimensional space. Using all the dimensions may lead to instability. Selecting the best attributes or dimensions may be a complex task, and is prone to overfitting. Thus, we propose an algorithm that clusters clauses based on a global measure of distance. Our method is based on the observation that if two examples are similar, they should be covered by the same clauses. In other words, two examples e_i and e_j are similar if when a clauses C covers e_i, it is likely that it will also cover e_j.

The Algorithm Next, we detail rkmeans. The algorithm proceeds in three steps. First, we compute *distances* between relational examples. Notice that we only compute distances between examples, we do not discretize examples into an attribute space. Given these distances, we then implement an agglomerative clustering algorithm. In this case, we apply the K-medoids algorithm, a clustering algorithm related to the K-means algorithm. The main difference between the K-medoids and K-means is the definition of centers: K-medoids uses data points (examples) as centers, the so-called *medoids*.

rkmeans proceeds in three steps:

- Preprocessing: Using the method described in [1, 3] represent each example $e \in$ E as a set of clauses $\mathcal{S}$ that can be deduced from e and B and that respect $\mathcal{L}$. This first step can be seen as a compilation of the raw data since it is done only once and can be used whenever data analysis is required.
- Distance Computation: Compute a distance between all pairs of examples e_i and e_j.
- Cluster Generation:
  1. Choose the number of clusters - k: The value can be user defined or automatically determined by performing an user-specified N clusterings for each $k \in k_m \ldots k_M$ and selecting the k that has best score through a method such as Silhouette validation [11].
  2. Generate the set of clusters C using the K-medoids algorithm;
  3. For each member of C output the clauses that potentially may describe the cluster.

Clusters Quality Criteria Many criteria have been developed for determining cluster quality (see e.g., [5, 6]). All have a common goal: find the clustering which results in compact clusters which are well separated. By default, rkmeans uses a straightforward measure that combines cluster compactness (bc) and separation (wc) by taking the ratio between the two, defined as: $quality(C) = bc(C)/wc(C)$. We want to minimize the distance within the clusters wc and maximize the distance between clusters bc, hence we want to minimize the quality measure. Therefore, the clustering which gives a minimum value for the quality measure will be considered the best.

Similarity of Structured Objects Next, we discuss in more detail the issues that arise when representing attributes of multi-relational objects, and justify the distance measure used. We denote the set of clauses that encodes an example a as S_a, and $|S_a|$ as the number of clauses in the set. The $\cap$ and $\cup$ denote, respectively, the operations of

intersection and union. Therefore, $S_a \cap S_b$ denotes a set that results from intersecting S_a and S_b, i.e., contains the clauses that occur in both sets.

We want to measure *similarity* between examples. Ideally, a measure of similarity should at least be symmetric, so that we can perform clustering. It should also be normalisable, so that we can compare distances between different examples.

We aim at proving that the measure should be based in what is common between sets of clauses, that is, on the shared attributes. Given S_a and S_b, this suggests it should be proportional to their intersection, say $S_a \cap S_b$. On the other hand, if $S_a \subset S_b$ the intersection will be the same as if $S_a = S_b$. It thus make sense to compare the intersection against both examples. A measure that achieves this goal is the Tanimoto coefficient:

$$tanimoto(a, b) = \frac{|S_a \cap S_b|}{|S_a \cup S_b|} = \frac{|S_a \cap S_b|}{|S_a| + |S_b| - |S_a \cap S_b|}$$

The Tanimoto measure is symmetric, normalised, and verifies the triangle inequality [9].

Naturally, several distance measures can be devised in the relational setting [10, 8]. In a way our proposal is similar to the RIBL [2] measure in the sense that both approaches calculate the distance between two objects/examples using their properties and other related objects (and respective properties) up to a certain depth. However, RIBL is limited to clauses function-free (e.g., lists and other compound terms were not supported). In [7] the RIBL similarity measure was extended to include lists and compound terms using edit-distance. Our proposal is not limited to function-free, is computed in a different way and, the clustering algorithm can still generate sets of clauses as a complement of the propositional model.

Changes to the Refinement Operator The method proposed to compute the distance between examples relies on recent work on encoding examples as trees that represent sets of clauses that are logical consequences of the example [1, 4]. Our motivation is to find all clauses that portray features for an example, and not the best overall clause. Therefore, it makes sense to filter clauses that do not contribute information as attributes. To understand these restrictions, it is convenient to see a clause as a graph, where literals are nodes, and variables embed directed edges between literals, such that the edge originates from the leftmost literal where the variable is an output variable. We support the following constraints:

1. The clause must form a single connected component; in other words, we should not be able to break the clause into two sub-clauses that do not share variables. If we can do so, then our clause is just a conjunction of two smaller clauses, and does not introduce any information that is not present in the join of the other attributes.
2. We do not allow clauses with *open edges*, i.e., if a variable is generated, it must be consumed. This restriction is based on the observation that most often such edges are used to *grow* the clause, by introducing new literals.

We shall also use a different restriction, widely used in ILP systems: we will generate all clauses up to a maximum length L. Our minimal length will be 2, the head literal and a literal in the body. Thus, if $L = 2$, then we are close to a straightforward

propositional approach. On the other hand, as we increase L we will be able to consider more attributes and look further, either on more detailed information about the example, or further away in the data-base.

Interpreting Clusters In general, a cluster may have more than a single explanation, i.e., which features of the examples can justify the cluster. Recall, that in a way, the clustering is made by aggregating the examples that have more clauses in common. Hence, arguably, the clauses over-represented may help, or even be sufficient, to understand a cluster. To do so, we look for clauses that have a different distribution in the cluster. A widely used way to estimate distances between distributions is the Kullback-Leibler (KL) divergence:

$$D_{KL}(P \parallel Q) = P\frac{log(P)}{log(Q)} + (1-P)\frac{log(1-P)}{log(1-Q)}$$

where Q is the probability of a clause being found in the whole data set and P is the probability that a clause is found in the cluster. Therefore, each cluster is represented by the clauses with higher KL divergence.

2 Experiments and Results

We experimentally address two main questions:

- is the data separated into distinct clusters?
- do we obtain a human understandble description/explanation of the clusters?

Table 1 briefly characterizes the data sets used in terms of number of examples and relations, and in terms of data-base size. Furthermore, for each data set is presented the average number of clauses generated by example, as a function of clause length. We chose these data sets because they are large enough to be quite challenging for traditional ILP learning, and because they address important applications that have a natural interpretation.

Data Set	$\mid E \mid$	$\mid T \mid$	$\mid V \mid$	$\mid R \mid$	$\mid B \mid$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$
Carcinogenesis	302	24k	48	30	47	151	3,770	54,762	530,334	–
Cora	1,295	22k	0	48	26	1,825	3,245	8,545	58,929	457,063
NCTRER	232	15k	18	12	41	25	1,316	12,755	85,022	469,249
Mutagenesis	171	16k	38	19	48	56	1,758	25,506	261,624	2,213,947

Table 1. Data Sets Characterization: $|E|$ is the number of examples; $\mid T \mid$ is the approximate total number of tuples (in thousands); $\mid V \mid$ is the approximate total number of rules/views defined; $\mid R \mid$ number of different rules/views; $\mid B \mid$ is the number of relations available to be included in the body of the generated rules; and $\mathcal{C}_i$ is the size of the universe of rules with up to i literals ($i = 2, 3, 4, 5, 6$).

There is no universal measure of cluster quality. We followed an often used approach to evaluate the clusters that involves assesing the level of agreement of the clusters with the true class labels available in all datasets. Note that class labels information is never used, in any way, during the clustering formation. The class label of a cluster corresponds to the majority class of its elements.

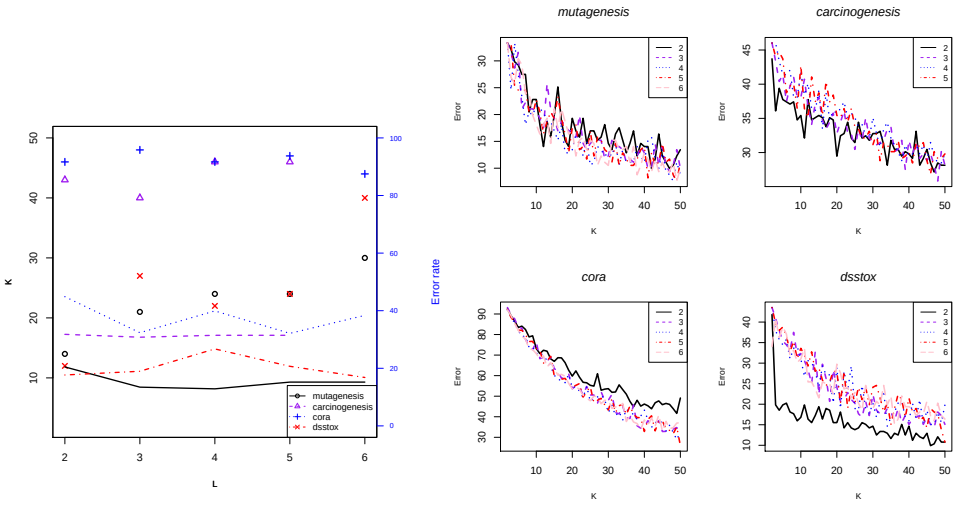

Fig. 1. A) Automatically determined K values (points) for different clauselengths (L) and respective clusters average error rate (lines).; B) average clusters error rate for different values of K and clauslengths (2, 3, 4, 5, and 6).

Naturally one would expect better accuracies as the number of clusters increases since with greater number of clusters one should find more smaller class pure clusters. In the limit, clusters with a single element will always be correct.

To study the sensitivity to its user-settable parameters we have performed a series of experiments varying the number of literals in the rules. Furthermore, to study the influence of the number of clusters we varied that number from 2 to 50 and compared it to the automatically determined K value.

Figure 1 summarizes the results for the best determined K values for different clause lengths. We highlight the following findings: i) increasing the value of L does not seem to affect clustering considerably; ii) the average cluster error achieved for the applications is very promissing, reaching in some cases, the best values reported in the literature.

As mentioned earlier, not just we can cluster examples, but quite often we find clauses that explain the clustering. For instance, one of the clusters generated for Mutagenesis encompasses 20 instances (all of class active). One of the rules proposed to describe this cluster is $active(A) : -bond(A, B, C, 2), phenanthrene(A, D)$, which is observed in all elements of the cluster and in more 7 instances.

3 Conclusion

We introduced and evaluated an approach for the clustering of relational instances where the distance between examples depends on the number of common clauses. Our

approach requires clause generation only as a preliminary step, and can scale up to large data-sets. We show that it can be used with excellent results for partitional clustering. Moreover, our technique can provide some insight into the data, by looking at clauses that have a very different distribution in the cluster. Such clauses may not necessarily explain the cluster, but they often provide valuable insight into why the examples cluster together. Note that although we have discussed partitional clustering, these same principles apply to hierarchical clustering and can be used to construct classifiers.

Can we further improve clustering? We would like to investigate why increasing L does not seem to affect clustering very much maybe. The advantages of deeper clauses may be offset by simply having more clauses. Therefore, it would make sense to experiment with even more restrictive approaches that we would further restrict acceptable clauses, such as disallowing clauses with independent *sub-components*. Other interesting approaches are to simply disallow categories of clauses, even short clauses, or to combine separate distances originating from different subsets of the original database.

Acknowledgment This work has been supported by Fundação para a Ciência e Tecnologia project HORUS (PTDC/EIA-EIA/100897/2008).

References

1. Rui Camacho, Nuno A. Fonseca, R. Rocha, and V. Santos Costa. Ilp :- just trie it. In *Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4894 of *LNAI*, pages 78–87. Springer-Verlag, 2008.
2. W. Emde and D. Wettschereck. Relational instance based learning. In *Proceedings 13th ICML*, pages 122 – 130. Morgan Kaufmann Publishers, 1996.
3. Nuno A. Fonseca, Rui Camacho, Ricardo Rocha, and Vitor Santos Costa. Compile the hypothesis space: do it once, use it often. *Fundamenta Informaticae*, Special Issue on Multi-Relational Data Mining(89):45–67, 2008.
4. Nuno A. Fonseca, R. Rocha, Rui Camacho, and V. Santos Costa. Ilp: Compute once, reuse often". In *6th Workshop on Multi-Relational Data Mining (MRDM 2007)*, 2007.
5. Julia Handl, Joshua Knowles, and Douglas B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.
6. Richard J. Hathaway and James C. Bezdek. Visual cluster validity for prototype generator clustering models. *Pattern Recogn. Lett.*, 24(9-10):1563–1569, 2003.
7. Tamas Horvath, Stefan Wrobel, and Uta Bohnebeck. Relational instance-based learning with lists and terms. *Machine Learning*, 43(1/2):53–80, April 2001.
8. Mathias Kirsten, Stefan Wrobel, and Tamas Horvath. Distance based approaches to relational learning and clustering. In *Relational Data Mining*, pages 213–232. Springer-Verlag, September 2001.
9. L. Ralaivola, S. J. Swamidass, H. Saigo, and P. Baldi. Graph kernels for chemical informatics. *Neural Netw.*, 18(8):1093–1110, 2005.
10. Jan Ramon and Maurice Bruynooghe. A framework for defining distances between first-order logic objects. In *ILP '98: Proceedings of the 8th International Workshop on Inductive Logic Programming*, pages 271–280, London, UK, 1998. Springer-Verlag.
11. Peter Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20(1):53–65, 1987.