An efficient and expressive similarity measure for relational clustering using neighbourhood trees

Sebastijan Dumančić and **Hendrik Blockeel**[1]

Abstract. Clustering is an underspecified task: there are no universal criteria for what makes a good clustering. This is especially true for relational data, where similarity can be based on the features of individuals, the relationships between them, or a mix of both. Existing methods for relational clustering have strong and often implicit biases in this respect. In this paper, we introduce a novel similarity measure for relational data. It is the first measure to incorporate a wide variety of types of similarity, including similarity of attributes, similarity of relational context, and proximity in a hypergraph. We experimentally evaluate how using this similarity affects the quality of clustering on very different types of datasets. The experiments demonstrate that (a) using this similarity in standard clustering methods consistently gives good results, whereas other measures work well only on datasets that match their bias; and (b) on most datasets, the novel similarity outperforms even the best among the existing ones.

1 Introduction

In relational learning, the data set contains instances with relationships between them. Standard learning methods typically assume data are i.i.d. (drawn independently from the same population) and ignore the information in these relationships. Relational learning methods do exploit that information, and this often results in better performance. Much research in relational learning focuses on supervised learning [2] or probabilistic graphical models [4]. Clustering, however, has received less attention in the relational context.

Clustering is an underspecified learning task: there is no universal criterion for what makes a good clustering, thus it is inherently subjective. This is known for i.i.d. data [3], and even more true for relational data. Different methods for relational clustering have very different biases, which are often left implicit; for instance, some methods represent the relational information as a graph (which means they assume a single binary relation) and assume that similarity refers to proximity in the graph, whereas other methods take the relational database stance, assuming typed objects that may participate in multiple, possibly non-binary, relationships.

In this paper, we propose a very versatile framework for clustering relational data. It views a relational dataset as a hypergraph with typed vertices, typed hyperedges, and attributes associated to the vertices. This view is very similar to the viewpoint of relational databases or predicate logic. The task we consider, is: cluster the vertices of one particular type. What distinguishes our approach from other approaches is that the concept of similarity used here is very broad. It can take into account attribute similarity, similarity of the relations an object participates in (including roles and multiplicity), similarity of the neighbourhood (in terms of attributes, relationships, or vertex identity), and interconnectivity or graph proximity of the objects being compared. We experimentally show that this framework for clustering is highly expressive and that this expressiveness is relevant, in the sense that on a number of relational datasets, the clusters identified by this approach coincide better with predefined classes than those of existing approaches.

2 Type-based clustering over neighbourhood trees

2.1 Hypergraph Representation

Relational learning encompasses multiple paradigms. Among the most common ones are the *graph* view, where the relationships among instances are represented by a graph, and the *predicate logic* or equivalently *relational database* view, which typically assumes the data to be stored in multiple relations, or in a knowledge base with multiple predicates. Though these are in principle equally expressive, in practice the bias of learning systems differs strongly depending on which view they take. For instance, *shortest path distance* as a similarity measure is much more common in the graph view than in the relational database view. In the purely logical representation, however, no distinction is made between the constants that identify a domain object, and constants that represent the value of one of its features. Identifiers have no inherent meaning, as opposed to feature values.

In this work, we introduce a new view that combines elements of both. This view essentially starts out from the predicate logic view, but changes the representation from a purely logical one to a hypergraph representation. Formally, the data structure that we assume in this paper is a typed, labelled hypergraph $H = (V, E, \tau, \lambda)$ with V being a set of vertices, and E a set of hyperedges; each hyperedge is an ordered set of vertices. The type function τ assigns a type to each vertex and hyperedge. The set of all vertex types is denoted as T_V, whereas T_E denotes the set of all hyperedge types. A set of attributes $A(t)$ is associated with each $t \in T_V$. The labelling function λ assigns to each vertex a vector of values, one for each attribute of $A(\tau(v))$. If $a \in A(\tau(v))$, we denote $a(v)$ the value of a in v.

Consider a knowledge base, with its Herbrand universe (the set of all constants) partitioned into domain constants and feature values. From this we can infer a typed, labelled hypergraph as follows. The set of vertices V equals the set of domain constants. We assume all constants are typed. Each vertex has a type $t(v)$ that equals the type of the domain constant. For each fact `p(d, v`$_1$`, v`$_2$`, ..., v`$_n$`)`, with d being a domain constant of type p and v_i feature values, there

[1] KU Leuven, Belgium, email: {sebastijan.dumancic, hendrik.blockeel}@cs.kuleuven.be

is a vertex d with type p and attribute vector $(v_1, \dots, v_m)$. For each fact `r(d1, d2, ..., dm)`, with d_i domain constants, there is a hyperedge $(d_1, d_2, \dots, d_m)$ with type r. Hyperedges are ordered sets, and do not have attributes[2].

The clustering task we consider is the following: given a vertex type $t \in T_V$, partition the vertices of this type into clusters such that vertices in the same cluster tend to be similar, and vertices in different clusters dissimilar, for some subjective notion of similarity. In practice, it is of course not possible to use a subjective notion; one uses a well-defined similarity function, which hopefully in practice approximates well the subjective notion that the user has in mind. To be able to capture several interpretations of relational similarity, such as attribute or neighbourhood similarity, we represent each vertex with a *neighbourhood tree* - a structure that effectively describe a vertex and its neighbourhood.

2.2 Neighbourhood tree

Consider a vertex v. A neighbourhood tree aims to compactly represent the neighbourhood of the vertex v and all relationships it forms with other vertices, and it is defined as follows. For every hyperedge E in which v participates, add a directed edge from v to each vertex $v' \in E$. Label each vertex with its attribute vector. Label the edge with the hyperedge type and the position of v in the hyperedge (recall that hyperedges are ordered sets). The vertices thus added are said to be at depth 1. If there are multiple hyperedges connecting vertices v and v', v' is added each time it is encountered. Repeat this procedure for each v' on depth 1. The vertices thus added are at depth 2. Continue this procedure up to some predefined depth d. The root element is never added to the subsequent levels.

2.3 Similarity measure

Given the definition above, comparing two vertices is achieved by comparing their neighbourhood trees. The main motivation behind the proposed similarity measure is to capture the distribution of different elements in the neighbourhood of a vertex, the elements being attributes of vertices in the neighbourhood, their identities and hyperedges. Given that in general a vertex participates in a non-fixed number of relations, where neighbours are described by (often overlapping) sets of attributes and their relations to other vertices, distributions are necessary to reliably model the neighbourhood of a vertex. Furthermore, comparing distributions avoids issues arising when comparing the vertices with very different number of neighbours (for example, properly normalizing for such scenario).

The similarity measure we propose relies on the similarity measure between multisets extracted from the corresponding neighbourhood tree. The multiset can be seen as features of the neighbourhood trees. For any neighbourhood tree g, let $B_l(g)$ be the multiset of vertices at depth l in g, and $B_{l,t}(g)$ the multiset of vertices of type t at depth l in g. All the vertices in $B_{l,t}$ have the same attributes, and each vertex assigns one value to each attribute; thus, for each attribute a, a multiset of values $B_{l,t,a}(g)$ is obtained. Let $B_{l,e}(g)$ be the multiset of labels of edges originating at vertices at depth l. Let $\mathcal{N}$ be the set of all neighbourhood trees corresponding to the vertices of interest in a hypergraph.

This way, the similarity is reduce to comparing different multisets. In principle, any measure over multisets of elements can be used, however, in our experiments, we chose to use the χ^2-distance between multisets [5], which is defined as:

$$d(A,B) = \sum_{x \in A \cup B} \frac{(f_A(x) - f_B(x))^2}{f_A(x) + f_B(x)} \qquad (1)$$

where A and B are multisets and $f_S(x)$ is the relative frequency of element x in multiset S (e.g., for $A = \{a, b, b, c\}$, $f_A(a) = 0.25$ and $f_A(b) = 0.5$).

The final similarity measure consists of a linear combination of different interpretations of similarity, reflected in the content of the multisets. Concretely, the similarity measure is a composition of components reflecting:

1. attributes of the root vertices,
2. attributes of the neighbouring vertices,
3. proximity of the vertices,
4. identity of the neighbouring vertices,
5. distribution of hyperedge types in a neighbourhood.

Each component is weighted by the corresponding weight w_i. These weights allow one to formulate an interpretation of the similarity between relational objects.

This formulation is somewhat similar to the *multi-view clustering* [1], with each of the components forming a different view on data. However, there is one important fundamental difference: multi-view clustering methods want to find clusters that are good in each view separately, whereas our components do not represent different views on the data, but different potential biases, which jointly contribute to the similarity measure.

2.4 Results

We compared the proposed similarity measure against a wide range of existing relational clustering approaches and graph kernels on five datasets. The proposed similarity measure was used in conjunction with spectral and hierarchical clustering algorithms. We found that, on each separate dataset, our approach performs at least as well as the best competitor, and it is the only approach that achieves good results on all datasets. Furthermore, the results suggest that decoupling different sources of similarity into a linear combination helps to identify relevant information and reduce the effect of noise.

Acknowledgements

Research supported by Research Fund KU Leuven (GOA/13/010)

REFERENCES

[1] Steffen Bickel and Tobias Scheffer, 'Multi-view clustering', in *Proceedings of the Fourth IEEE International Conference on Data Mining*, ICDM '04, pp. 19–26, Washington, DC, USA, (2004). IEEE Computer Society.

[2] Luc De Raedt, *Logical and relational learning*, Cognitive Technologies, Springer, 2008.

[3] Vladimir Estivill-Castro, 'Why so many clustering algorithms: A position paper', *SIGKDD Explor. Newsl.*, **4**(1), 65–75, (June 2002).

[4] Lise Getoor and Ben Taskar, *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*, The MIT Press, 2007.

[5] Haifeng Zhao, Antonio Robles-Kelly, and Jun Zhou, 'On the use of the chi-squared distance for the structured learning of graph embeddings', in *Proceedings of the 2011 International Conference on Digital Image Computing: Techniques and Applications*, DICTA '11, pp. 422–428, Washington, DC, USA, (2011). IEEE Computer Society.

[2] A fact of the form `p(d1, ..., dn, v1, ..., vm)` can always be written as `r(d1, ..., dn, h)` and `q(h, v1, ..., vm)`, where **h** denotes a *hyperedge identity*