Clustering Relational Data Using Attribute and Link Information

Jennifer Neville, Micah Adler, David Jensen
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts,
140 Governors Drive, Amherst, MA 01003 USA
{jneville | micah | jensen} @cs.umass.edu

Clustering is a descriptive task that seeks to identify natural groupings in data. Relational data offer a wealth of information for identifying groups of similar items. Both attribute information and the structure of relationships can be used for clustering. Graph partitioning and data clustering techniques can be applied independently to relational data but a technique that exploits both sources of information simultaneously may produce more meaningful clusters. This paper will describe our work synthesizing data clustering and graph partitioning techniques into improved clustering algorithms for relational data.

1 Introduction

Clustering is a descriptive task that seeks to identify natural groupings in data. Developing techniques to automatically discover such groupings is an important part of knowledge discovery and data mining research. The majority of data routinely captured by businesses and organizations are relational in nature yet few clustering techniques have been developed to take advantage of both the attribute information and the structure of relationships in relational data. The aim of our research is to synthesize work in data clustering and graph partitioning to improve clustering in relational data.

Relational data consist of objects (representing people, places, and things) connected by links (representing persistent relationships among objects). For example, relational data could be used to represent the motion picture industry, where objects represent studios, movies, and persons (e.g., actors, directors, and producers) and links represent relationships (e.g., actor-in and remake-of). Clustering algorithms may be used in isolation to describe the data in a set of higher-level patterns (i.e. the clusters themselves). Clustering may also be included as a component in a larger knowledge discovery system. In this case, cluster techniques may be useful to create new attributes for learning predictive models. For example, clustering actors in the movie data may produce groupings that represent abstract types such as *action-hero* or *teenage-idol*. Actor *type* could then be used to improve predictions of a movie's box office success.

Conventional data clustering algorithms identify groups of similar items in a dataset based on their attribute values. For example, actors would be clustered based on their age, gender and nationality. Data clustering algorithms vary significantly, with different notions of "similarity" and diverse approaches to construction of the clusters. However little work in data clustering focuses on relational data.

Traditional graph partitioning algorithms use the structure of a graph to find highly connected components. These approaches focus on the organization of edges in the graph and assign the nodes to a set of clusters in such a way that prescribed properties such as minimum cutsize or maximum connectivity are optimized. For example, actors may be clustered by the edges that represent *starred-with* relationships to group people into sets of actors that star in many movies together. Graph partitioning techniques were developed for use on graphs where most of the information is contained in the structure — edge or node weights is the only attribute information that has been considered.

2 Clustering Relational Data

Conventional clustering algorithms use attribute information to group examples under the assumption that two instances are related if they have similar attribute values. However, relational data have more information available to disambiguate groupings. We hypothesize that links confer a relationship between two instances in the same way that similar attribute values indicate a relationship. As such, clustering algorithms that incorporate link information should be able to produce better groupings. Co-citation analysis (Small and Giffith 1974) is based on a similar hypothesis — if many pages point to a set of pages, then the set pages are likely to address the same topic. Likewise, if a set of pages all point to the same pages, then the set of pages are likely to be semantically related.

Both data clustering and graph partitioning techniques can be used to cluster relational data — relationships in the data provide graph structure and attribute values on objects and links provide data information. Each approach used independently may offer insight into the data but there are also potential benefits to examining attribute and relations at the same time.

First, attributes and structure can be used to cluster for objects playing similar roles in the data. Clusters such as these identify groups of items that are similar both in their attributes and their relations to other types of instances. For example, *leading-ladies* have similar gender and salary attributes and also star in many *blockbuster* movies.

Second, attributes and structure can be used to cluster for communities in the data. Community clusters identify groups of items that have similar attributes and are also highly inter-connected. For example in citation data, a group of papers with similar terms and many intra-group citations may indicate an emergent research topic. In genomic data, a group of genes with similar attributes and many common interactions may all be involved in a similar function in the cell.

This work is focused on finding communities in relational data. The underlying assumption is that there is a latent (hidden) cluster variable for each object that influences both the attribute values intrinsic to the object and its relationships to other objects. In particular, objects are more likely to link to other objects within the same cluster than objects in other clusters and objects within a cluster are more likely to have similar attribute values than objects in different clusters.

Given noise-free data generated from the underlying process described above it should be possible to recover the cluster structures using either data clustering or graph partitioning alone. However, noise in either the attribute values or the link structure could reduce the accuracy of clusterings formed from only a single source of information. In the presence of noisy data, an algorithm that examines both structure and attributes simultaneously should be able to combine both sources of information to improve its clustering results.

The noise discussed above is assumed to be a random process, however, it is also possible that there are many alternative groupings for the same set of relational data. In this case, there would be multiple hidden cluster variables, each influencing a set of attribute values and relationships. An algorithm would need to select a relevant set of attributes and links before clustering in order to prevent the alternative clusterings from rendering the groups indistinguishable. Examining both structure and attributes simultaneously may provide more information for the selection process and result in improved clusterings.

3 Conventional Techniques

Conventional data clustering algorithms identify groups of similar instances in a dataset based on their attribute values (Arabie et al. 1996). There are many different measures of association used for clustering, both general and domain-specific. Instances can be represented as points in multi-dimensional attribute space where Euclidean distance is used as a metric. Other common metrics are variations of the standard cosine similarity measure or simple matching measures such as Dice's coefficient and Jaccard's coefficient.

K-clustering algorithms partition the instances into *k* disjoint groups. Hierarchical clustering algorithms produce a dendogram of clusters, where the lowest-level clusters each consist of a single instance, and all other clusters are made up of a set of smaller clusters. Divisive algorithms take a top-down approach to clustering; all items start in a single cluster and further clusters are formed from recursively splitting clusters into smaller components. Agglomerative algorithms take a bottom-up approach; all items start off in separate clusters and successive clusters are formed from merging sets of smaller clusters.

Traditional graph partitioning algorithms use the structure of a graph to find highly connected components (subgraphs) (Alpert and Kahng 1995). Given a graph $G=(V,E)$ the algorithms assign the vertices V to a set of k partitions (clusters) in such a way that prescribed properties such as minimum cutsize or maximum connectivity are optimized. Graph partitioning techniques were developed for use on graphs where most of the information is contained in the structure — edge and node weights are the only attribute information that is considered.

The general goal is to partition the graph such that connections within clusters are maximized and connections between clusters are minimized. When k is small (e.g. $k \leq 15$), this is usually referred to as the *partitioning* problem and top-down algorithms are used. When k is large, (eg. $k=O(|V|)$) this is usually referred to as the *clustering* problem and bottom-up algorithms are used. Partitioning techniques are often designed to operate on $V \times V$ matrix of edge weights. These techniques can be used for data clustering problems providing the data are represented as an $N \times N$ matrix of similarity scores (entry n_{ij} is the similarity of instances i and j). In this situation, the data would form a complete graph — every pair of instances would have an edge between them.

Both data clustering and graph-partitioning techniques can be used to cluster relational data — relationships provide graph structure and attribute values provide data information. This paper will examine three adaptations of existing techniques to consider both link structure and attribute information. Each approach modifies a graph-partitioning algorithm to consider both link structure and attribute similarity by weighting the existing link graph by similarity scores.

4 Hybrid Techniques

We investigate methods of adapting conventional techniques to incorporate both link structure and attribute information. Each approach uses an existing graph-partitioning algorithm, including attribute information by weighting the existing link graph with an attribute similarity metric.

4.1 Similarity Metric

A similarity function ***S*** defines the similarity between each pair of objects in a graph $G=(V,E)$. The similarity between objects i and j is determined by examining each of k attributes on the two objects and counting the number of attribute values they have in common. Objects that are not directly related by an edge in the graph have a similarity of 0 regardless of their attribute values.

$$S_{ij} = \begin{cases} \sum_k s_k(i,j) & \text{if } e_{ij} \in E \text{ or } e_{ji} \in E \\ 0 & \text{otherwise} \end{cases}$$

$$s_k(i,j) = \begin{cases} 1 & \text{if } k_i = k_j \\ 0 & \text{otherwise} \end{cases}$$

This metric is generally known as the *matching coefficient*. The similarity measure is used to weight the edges of the graph *G*.

4.2 Karger Min-Cut

Karger's Min-Cut algorithm (1993) finds the minimum cut of an undirected, edge-weighted graph. It is a Monte Carlo randomized algorithm that simplifies the standard deterministic approach. The basic algorithm is as follows:

1. Start with each vertex in its own cluster.
2. Pick a random inter-cluster edge, where the probability of selecting an edge is determined by the edge weights.
3. Merge the clusters connected by the selected edge.
4. When only two clusters remain, return the remaining inter-cluster edges as the cut.

In order to improve the likelihood of finding the minimum cut, this procedure is repeated multiple times and the best cut is returned. This cut partitions the graph into two clusters. Karger's algorithm can be used as a divisive, hierarchical clustering algorithm by applying the algorithm recursively on each partition until each cluster consists of a single node.

4.3 MajorClust

MajorClust is a simple and intuitive *k*-clustering algorithm designed for unweighted graphs (Stein and Niggemann 1999). Initially, the algorithm assigns each node of the graph to its own cluster. The algorithm then proceeds iteratively. On each iteration, a node moves to the cluster that is most prevalent among its neighbors. If there exist several such clusters, one of them is chosen randomly. The algorithm terminates when the clustering has stabilized (i.e. every node remains in the same cluster). MajorClust can easily be extended to consider edge weights by modifying the selection of most prevalent clusters appropriately.

4.4 Spectral

Spectral partitioning algorithms use a weighted adjacency matrix to cluster connected graphs. We base our approach on previous work by Shi and Malik in image segmentation (2000). Shi and Malik developed a divisive, hierarchical clustering algorithm that uses spectral partitioning with a normalized cut objective function. Each level of recursion finds a partition (A,B) of the nodes V that minimizes the normalized cut objective function $J(A,B)$ subject to the constraints that $A \cap B = \emptyset$ and $A \cup B = V$:

$$J(A,B) = \frac{cut(A,B)}{\sum_{i \in A} d_i} + \frac{cut(A,B)}{\sum_{j \in B} d_j}$$

$$cut(A,B) = \sum_{i \in A,\, j \in B} S_{ij}$$

$$d_i = \sum_k S_{ik}$$

This optimization problem can be cast as an eigenvector problem using one additional matrix ***D*** – a diagonal matrix of weighted degrees d_i (defined above). The eigenvector corresponding to the second smallest eigenvalue of the generalized eigensystem $(\boldsymbol{D}\text{-}\boldsymbol{W})y=\lambda \boldsymbol{D}y$ is a continuous solution minimizing $J(A,B)$. The eigenvector values of this solution can be used to partition the nodes of the graph. Because the solution is continuous, it is necessary to pick a threshold to split the values into two partitions. N evenly spaced points over the range of values are examined and the split that minimizes $J(A,B)$ is chosen. The algorithm is applied recursively to each of the resulting partitions.

5 Synthetic Data Experiments

It is known to be a difficult task to evaluate clustering on datasets for which there is no "right" answer (Arabie et al. 1996). Before applying these hybrid algorithms on real-world relational data we will examine the effectiveness of these three algorithm on synthetic data sets for which the correct clustering is known. This facilitates measurement of algorithm performance over a wide range of conditions. From this study we hope to identify the situations under which we expect each of the clustering algorithms to perform well.

5.1 Methodology

Our synthetic data sets are comprised of undirected, unipartite, connected graphs. Each graph contains 200 nodes (objects). A binary attribute, C={+,-} is used to represent cluster membership for each object. The clusters are assigned randomly with $P(+)=0.5$. Each object has five binary attributes whose values are determined by the object's cluster label (e.g. $P(A_5=1|C=+)=0.9$ and $P(A_5=1|C=-)=0.1$). Cluster labels determine which edges (links) are added to the graph. Edges are generated in a probabilistic manner, each of the possible V^2 edges are added with $P(e_{ij}|C_i=C_j)=p_1$ and $P(e_{ij}|C_i \neq C_j)=p_2$.

The goal of this work is to use attribute and link information to improve clustering results. The implicit assumption is that an approach using both sources of information will do better than an approach using either

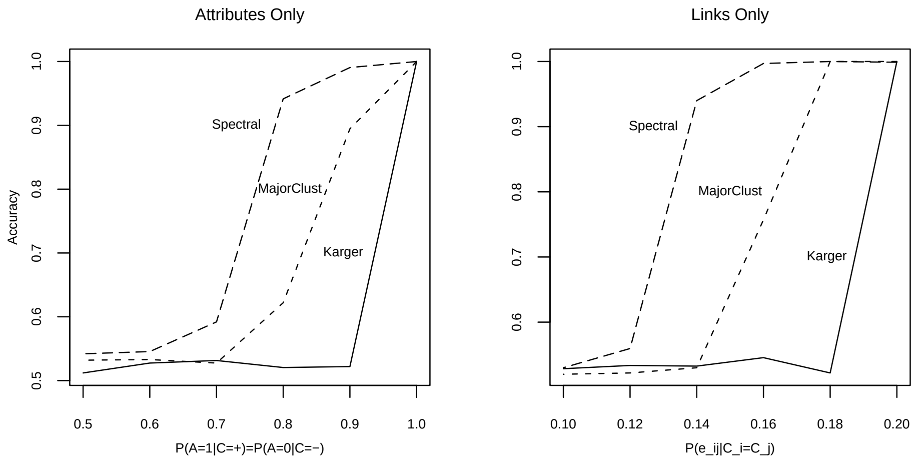

Figure 1: Algorithm accuracy when using attribute information and link structure in isolation.

source in isolation. To evaluate this claim, we record performance of each algorithm using (1) attribute information only, and (2) link information only.

The experiments considering attribute information in isolation used a complete graph where each pair of objects is connected. These experiments varied the strength of the relationship between the attribute values and the cluster label (see range below). Within each level of correlation, all five attributes are generated with the same probability:

$P(A=1|C=+)= P(A=0|C=-)=[0.5,1.0]$

$P(A=0|C=+)= P(A=1|C=-)=1- P(A=1|C=+)$

The experiments considering link information in isolation used the existing relationship graph with uniform edge weights, set to 1. These experiments varied the strength of the relationship between the link structure and the cluster label. Intra-cluster and inter-cluster links are generated with the following range of probabilities:

$P(e_{ij}|C_i=C_j)=[0.1,0.2]$

$P(e_{ij}|C_i \neq C_j)= 0.2 - P(e_{ij}|C_i=C_j)$

This range of probabilities was chosen to result in a graph with approximately 2000 edges. This level of linkage is comparable to the levels we have observed in real-world relational data sets.

The third set of experiments record algorithm performance while varying both attribute and link association. These experiments use the existing graph structure, weighted by similarity. The ranges of attribute and link association are again:

$P(A=1|C=+)= P(A=0|C=-)=[0.5,1.0]$

$P(A=0|C=+)= P(A=1|C=-)=1- P(A=1|C=+)$

$P(e_{ij}|C_i=C_j)=[0.1,0.2]$

$P(e_{ij}|C_i \neq C_j)= 0.2 - P(e_{ij}|C_i=C_j)$

All experiments report the accuracy of the clusterings returned by each algorithm. Accuracies are averaged over ten trials at the same settings.

5.2 Results

Figure 1 shows the results of the first two experiments. The spectral algorithm performs well over a wider range of data sets than either of the other two algorithms. Remember that the spectral algorithm is optimizing a different criterion than the Karger algorithm. This may indicate that normalized cut is a better function to partition graphs with low linkage. Min-cut will often choose an imbalanced partition with a small number of nodes with low degree on one side and the rest of the graph on the other. Normalized cut penalizes small clusters in the hopes of finding a more balanced partition.

Figure 2 shows the results of the third experiment where the strength of both attribute and link correlations are varied simultaneously. The results are similar to the first two experiments in that the spectral algorithm performs best over a wide range of data sets. From these graphs we can see that the hybrid spectral algorithm sometimes does not do as well as the spectral algorithm that examines attribute/link information in isolation. In particular, consider the attribute(link)-only results where the association with the cluster is moderate. When the attribute (link) association is moderate and the link (attribute) association is low the hybrid spectral algorithm achieves significantly lower accuracy than would be achieved considering only attributes (links) in isolation. This indicates a potential damaging effect of using the additional information.

6 Related Work

Probabilistic models have been developed to model cluster membership using both attribute information and link structure. Cohn and Hoffman (2001) outline a generative model where a document's topic determines both its content and its citations. The model without link structure (content only) is known as probabilistic latent semantic indexing (pLSI). To our knowledge, the Cohn and Hoffman model has not been evaluated in a cluster-

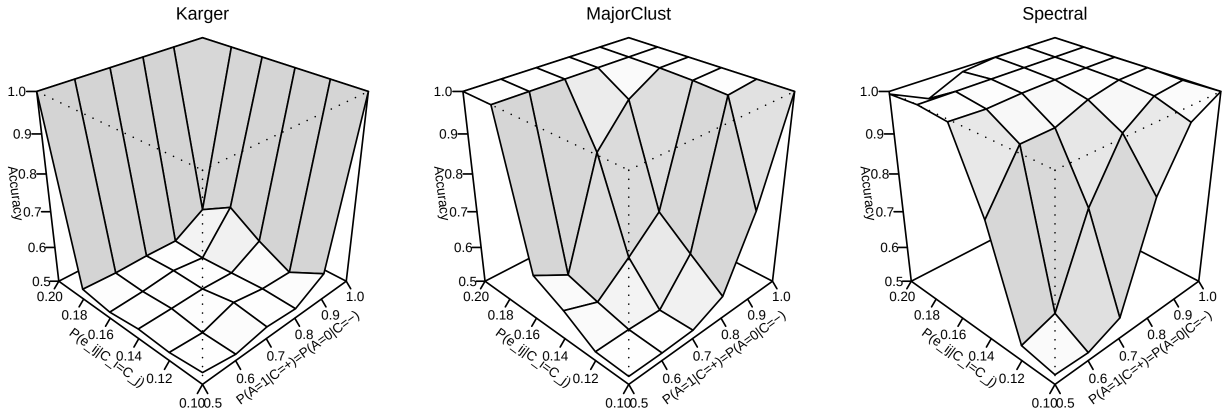

Figure 2: Algorithm accuracy for varying levels of attribute value and link structure correlation with cluster membership.

ing context. Kubica, Moore, Schneider, and Yang (2002) propose a probabilistic model of link structure based on cluster membership. The model considers both attribute information and link structure but combines them in an alternative manner. In the generative model, attributes determine group membership and group membership determines the link structure. Taskar, Segal and Koller (2001) use probabilistic relational models (PRMs) to cluster relational data with attribute and links. PRMs are directed graphical models, which can be used to cluster for hidden group variables. However, the model's acyclicity constraint makes it difficult to apply to network data with complex dependencies.

HyPursuit (Weiss et al. 1996) was the first information retrieval system to cluster documents using semantic information in both document contents and hyperlink structure. The system defined a complex similarity metric to capture both content and link structure correspondence between pages. The hybrid similarity metric can be used with any conventional clustering algorithm because it is defined over all pairs of pages. It is difficult to assess the utility of the metric however, because evaluation consists of a subjective assessment on a single clustering task.

Modha and Spangler (2000) also propose an algorithm for clustering hypertext documents using document contents and hyperlink structure. The authors capture three features of documents in their new similarity measure: (1) word similarity, (2) out-link similarity, and (3) in-link similarity. A geometric hypertext-clustering algorithm is used, which extends the classical Euclidean k-means algorithm (Everitt 1993). Modha and Spangler include parameters to control the influence of the three features. They include a search for the optimal parameter setting in their algorithm but do not evaluate the impact of different settings. They do note that several settings were chosen across clustering experiments. This indicates that different web graphs may contain varying levels of textual and link information.

He, Ding, Zha, and Simon (2001) use a spectral graph-partitioning algorithm to automatically identify topics in sets of retrieved web pages. This approach is quite similar to our spectral approach, however He et al. use a different similarity measure designed for high-dimensional text domains. In addition, they augment the hyperlink graph with weighted co-citation links. The algorithm automatically clusters query result sets for topics and presents the user with the most authoritative pages from each topic.

7 Future Work

This paper presents our preliminary work in clustering algorithms that exploit both attribute information and link structure to improve clustering of relational data. There are many questions still to be explored.

Are the algorithms robust to irrelevant attributes/links? Our initial experiments in this area indicate that the spectral algorithm is surprisingly robust to irrelevant attributes. Figure 2 indicates that the algorithm relies more heavily on link information (i.e. it performs worse when link information is ambiguous than when attribute information is ambiguous). This may indicate that the spectral algorithm will be less robust to irrelevant links.

Can the similarity metric be improved? Varying the relative influence of the attribute and link information may improve algorithm performance over a wider range of data sets. This is evidenced by the discrepancies between the results in figure 1 and 2. A hybrid algorithm should perform at least as well as an algorithm that has only attribute or link information available.

How far influence should travel in the relationship graph? It seems plausible that objects more than one link away still contain useful disambiguating information. However, indirect relationships may need to be modulated by attribute information — inter-cluster neighbors are unlikely to help clustering and are more likely to drive the clusters farther a field.

In conclusion, it is intuitively plausible that link structure can be combined with attribute information to effectively group relational data. We have set up a framework to evaluate algorithms and similarity metrics quantitatively over a wide range of relational data sets. We have used this framework to develop clustering algorithms that effectively exploit both sources of information. The performance of the spectral algorithm is promising, but there are still areas for improvement. Our current work is focused on understanding the effects of algorithm choices and data characteristics on algorithm performance and applying these algorithms to real-world datasets.

References

Alpert, C. J. and A. B. Kahng. (1995) Recent Directions in Netlist Partitioning: A Survey. Integration, the VLSI Journal.

Arabie, P., L. Hubert and G. DeSoete. (1996) Clustering and Classification. World Scientific.

Cohn, D. and T. Hofmann. (2001) The missing link - a probabilistic model of document content and hypertext connectivity. In Advances in Neural Information Processing Systems, Vol. 10.

Everitt, B. (1993) Cluster Analysis. John Wiley & Sons, Inc.

He, X., C. Ding, H. Zha, and H. Simon. (2001) Automatic topic identification using webpages clustering. Proc. IEEE Int'l Conf. Data Mining. San Jose, CA, pages 195--202.

Karger, D.R. (1993). Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 21--30.

Kubica, J., A. Moore, J. Schneider, and Y. Yang. (2002) Stochastic link and group detection. Proceedings of the Eighteenth National Conference on Artificial Intelligence, pp. 798-804.

Modha, D. and W. Spangler. (2000) Clustering hypertext with applications to web searching. In Proceedings of the 11th ACM Conference on Hypertext and Hypermedia, pages 143--152.

Shi, J. and J. Malik. (2000) Normalized cuts and image segmentation. IEEE Trans. Part. Anal. Mach. Intell., 22(8):888-905.

Small, H. and B. Griffith. (1974) The structure of scientific literatures I: Identifying and graphing specialties. Science Studies, vol. 4, pp.17-40.

B. Stein and O. Niggemann. (1999) On the nature of structure and its identification. 25th Workshop on Graph Theory, Lecture Notes in Computer Science, Springer-Verlag.

Taskar, B., E. Segal, and D. Koller. (2001) Probabilistic clustering in relational data. In Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01), pages 870-87.

Weiss, R., B. Velez, M. Sheldon, C. Namprempre, P. Szilagyi, A. Duda and D. Gifford. (1996) HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering. ACM Conference on Hypertext, Washington USA.