Data Clustering with a Relational Push-Pull Model

Adam Anthony and Marie desJardins
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250
{aanthon2, mariedj}@cs.umbc.edu

Abstract

We present a new generative model for relational data in which relations between objects can have either a binding or a separating effect. For example, in a group of students separated into gender clusters, a "dating" relation would appear most frequently between the clusters, but a "roommate" relation would appear more often within clusters. In visualizing these relations, one can imagine that the "dating" relation effectively pushes *clusters apart, while the "roommate" relation* pulls *clusters into tighter formations. A unique aspect of the model is that an edge's existence is dependent on both the clusters to which the two connected objects belong and the features of the connected objects. We use simulated annealing to search for optimal values of the unknown model parameters, where the objective function is a Bayesian score derived from the generative model. Results describing the performance of the model are shown with artificial data as well as a subset of the Internet Movie Database. The results show that discovering a relation's tendency to either push or pull is critical to discovering a consistent clustering.*

1. Introduction

Relational data clustering is a form of relational learning that clusters data using the relational structure of data sets to guide the clustering. Many approaches for relational clustering have been proposed recently. The common assumption in much of this research is that *relations have a binding tendency* [6, 1, 8, 7]. That is, edges are assumed to appear more frequently within clusters than between clusters.

This binding quality may be too strong an assumption. Bhattacharya and Getoor [1] acknowledge that it is possible for a relation to provide "negative evidence," where the presence of a relation between two objects implies that the objects belong in different clusters. If most of the edges in a relational set provide negative evidence, then this set should be considered to have a *separating*, rather than a binding, tendency. Consider a social network of university students containing both men and women. If the social network is partitioned by gender, edges for a dating relation will appear most frequently between the clusters. This can be visualized as an approximately bipartite structure between the two clusters that "pushes" them apart. Edges for a roommate relation, on the other hand, will appear most frequently within the clusters. This binding relation can be visualized as a net that "pulls" the objects in a cluster closer together.

Given that a relation can have either of these two tendencies, accurate clustering in relational data requires determining whether each relation tends to bind or separate objects. Bhattacharya and Getoor [1] and others have used domain-specific knowledge to identify and process negative evidence. The contribution of our research is a domain-independent model that handles the case where the tendency of each relation is used to guide the clustering. These tendencies can be provided as background knowledge, or, if such knowledge is unavailable, they can be inferred from the data using an estimation method that we present.

2. The Relational Push-Pull Model

We propose the Relational Push-Pull Model (RPPM), a Bayesian model that can be used to find clusterings in relational data. In this model, object features depend on the cluster the object belongs to, modeled as a mixture of probability distributions (e.g., a mixture of Gaussians), where the objects in each cluster are assumed to be generated by the distribution specified for that cluster. To formulate a relation distribution, we use the concept of *existence uncertainty* [3] for relations. For each pair of objects l and m in a data set, we calculate the probability that a link exists between the objects. For the remainder of this paper, it is assumed that the data is represented as a relation graph where the objects are vertices and there is labeled edge between two objects if a relation of a certain type exists between them.

Ultimately, the goal is to determine the most likely clustering, given the set of observed objects (O) and the set of observed edges associated with the relations (R). The probability of a clustering, $Pr(C \mid O, R)$, can be used as an objective function for a local search method, and can be computed using Bayes' rule as:

$$Pr(C \mid O, R) = \alpha\, Pr(R \mid C, O) Pr(C \mid O), \tag{1}$$

where α is a normalizing constant. The remainder of this section explains how we calculate the two right-hand terms: $Pr(R \mid C, O)$ and $Pr(C \mid O)$.

2.1. Modeling Relation Graphs as Push or Pull

Getoor et al. [3] present an edge existence probability that is somewhat different from ours. In their work, edge existence depends only on the class labels of the objects that the edge would connect. The notion of edge existence in an RPPM also incorporates the concept of *relational autocorrelation* [4]: that is, the tendency for feature values in linked objects to have similar values.

The advantage of the RPPM is that it can be used to quantitatively represent both push and pull relations as a function of both cluster membership and the object's features. To explain how this is achieved, we first discuss the benefits of a cluster-only edge existence model and a feature-based relational autocorrelation model separately, then show how the two ideas can be combined.

One way to model edge existence using only cluster labels is to specify two constant probabilities, IC and OC. IC (*in cluster*) is the probability of an edge existing between l and m if the objects are in the same cluster, and OC (*out cluster*) is the probability that the edge exists if they are in different clusters. Pull-type relation graphs are created when the value of IC is much greater than OC. Conversely, push-type relation graphs result when the value of IC is much smaller than OC. In either case (push or pull), edge information can easily be used to help find the clustering. When IC is very similar to OC, the relation graph's tendency is ambiguous, and cannot help find the clustering. This is analogous to clustering evenly-distributed, sparse data objects: there are no patterns in the data to exploit. The benefit of using constants to model edge existence is that it clearly models whether a relation's tendency is push or pull. The drawback is that it makes no use of relational autocorrelation.

Jensen and Neville [4] claim that relational autocorrelation is a common phenomenon in relational data sets. A simple approach to exploiting autocorrelation for edge existence is to use a probabilistic function of the objects' features, $f(l, m)$, where l and m are objects, represented as feature vectors. Using a function of object features for edge existence has the advantage of emphasizing the influence of autocorrelation, but it does not model the relation graph's tendency to push or pull.

RPPM incorporates the benefits of both approaches by making edge existence dependent on both cluster membership and object features. This dependence is modeled by the following formula:

$$\begin{aligned} & Pr(e_{lm} \mid l \in C_i, m \in C_j, \lambda_{I_t}, \lambda_{O_t}) \\ & \quad = \begin{cases} IC := f(l,m) - (1 - \lambda_{I_t}) & \text{if } C_i = C_j; \\ OC := f(l,m) - (1 - \lambda_{O_t}) & \text{if } C_i \neq C_j. \end{cases} \end{aligned} \tag{2}$$

Instead of being constant, IC and OC are functions that are selected depending on the cluster membership of l and m. The parameters λ_{I_t} and λ_{O_t} determine which of the two functions is higher for a relation type t, and thus whether intra-cluster edges or inter-cluster edges are more likely. The expression $(1 - \lambda)$ is used above to make the value of each lambda more intuitive: a higher lambda corresponds to a higher probability of edge existence. If an IC or OC value becomes less than zero, the probability value for that edge is constrained to be zero, so that all values returned are nonnegative. Equation 2 assumes that each cluster has the same IC and OC functions..

To add support for directed edges, we introduce a new parameter, Λ_t, which is a $k \times k$ matrix (where k is the number of clusters) that replaces λ_{I_t} and λ_{O_t} for a relation type t. The diagonal entries in Λ_t are the in-cluster parameter values, so that the probability of an edge existing between objects l and m that are both in cluster 1 is determined using the value stored in Λ_{t11}. On either side of the diagonal, the $(i, j)^{\text{th}}$ entry in Λ_t specifies the probability that a directed edge exists between a randomly selected object in cluster i and a randomly selected object in cluster j. Using Λ_t, Equation 2 can be modified to support directed edges as follows:

$$Pr(e_{lm} \mid l \in C_i, m \in C_j, \Lambda_t) = f(l,m) - (1 - \Lambda_{tij}) \ . \tag{3}$$

The probability distribution of a relational graph for one edge type (G_t) is the product of each edge's probability of existence:

(4)
$$Pr(G_t \mid C, O) = \prod_{e_{lm} \in G_t} Pr(e_{lm} \mid \ldots) \times \prod_{e_{lm} \in O \times O, \notin G_t} \{1 - Pr(e_{lm} \mid \ldots)\} \ ,$$

where $Pr(e_{lm} \mid \ldots)$ is calculated as in Equation 2. The conditioning contexts are omitted to improve readability.

Finally, the probability distribution of the entire relation space can be calculated as the product of the distribution of each relation graph with type t:

(5)
$$Pr(R \mid C, O) = \prod_{\text{all t}} Pr(G_t \mid C, O),$$

which completes the definition of the first term in Equation 1.

2.2. Modeling Object Features

As stated previously, the feature space is modeled as a mixture of distributions. Under this model, if an object is assigned to a cluster, its features are assumed to be sampled from the distribution specified by that cluster's parameters. The second term of Equation 1 is computed as:

(6)
$$Pr(C \mid O) = \alpha \, Pr(O \mid C) \times Pr(C)$$

Where $Pr(O \mid C)$ is derived using the distribution parameters and $Pr(C)$ is computed as a multinomial distribution.

2.3. Learning an RPPM Model

We are currently using simulated annealing to evaluate the model's potential. Simulated annealing is a better choice than a more basic hill-climbing approach because of the tradeoff between searching in the feature space and searching in the relation space. It is possible that a maximization in one space will reduce the term in Equation 1 for the other space. This results in many local maxima, making simulated annealing an appropriate choice. In Section 3.1, we show empirically that simulated annealing can find the correct clustering.

There are two open-ended components of the model that are domain-specific: the edge existence function $f(l, m)$ and the feature model distribution. The experiments in this paper assume a mixture of Gaussians for the feature model.

For the edge existence function, an intuitive representation is to model an edge e_{lm}'s existence probability as a decreasing function of normalized Euclidean distance (δ_{lm}). This models the situation where objects similar in the feature space are more likely to be related. For example, people who are closer in age are more likely to be roommates. Given this assumption, we chose the following function for edge existence probability:

(7)
$$Pr(e_{lm} \mid l \in C_i, m \in C_j, \Lambda_t) = e^{-\delta_{lm}} - (1 - \Lambda_{tij}) \ .$$

We use a linear cooling schedule $T = 1 - \frac{t}{R}$, where t is the elapsed time and R is a sufficiently large value (set to 10,000 in the experiments in Section 3) that controls the rate at which the temperature decreases. At each time step t, the simulated annealing algorithm generates a successor in the following way.

1. Select $k \times \tau$ objects uniformly from the feature space as candidates for moving. τ is an integer ≥ 1 that controls the number of objects selected.
2. For each selected candidate, choose a cluster to assign it to randomly with uniform probability—it may be assigned to the same cluster it is currently assigned to.
3. For each cluster k, compute the maximum likelihood estimates for the feature and edge model parameters.

2.4. A Lambda Matrix Estimator

To estimate the lambda values in Equation 3, RPPM only needs to calculate the average value of $f(l, m)$ for each pair of objects (l, m), where $l \in C_i$ and $m \in C_j$. Then, let c be the number of observed edges from C_i to C_j, and n be the total number of *potential* edges between C_i and C_j ($n = |C_i| * |C_j|$). A reasonable estimate for the probability that an edge exists between C_i and C_j is $\frac{c}{n}$. If the expected edge existence value $\hat{f}(l, m)$ is known, it is straightforward to solve for Λ_{tij}:

$$\frac{c}{n} - \hat{f}(l, m) + 1 = \Lambda_{tij} \ .$$

There are cases where this estimator can produce Λ_t entries that are not within the bounds of $0 \leq \Lambda_{tij} \leq 1$. This is because of the variable nature of the function f and the relatively constant nature of $\frac{c}{n}$. In such cases where the estimator computes an inconsistent value, we constrain the value to be either one or zero, depending on which boundary is violated.

3. Experimental Results

This section presents results from two experiments, one using artificial data and one that analyzes a subset of the Internet Movie Database. Where accuracy measurements are reported, they refer to the average number of objects clustered correctly. Using pre-labeled data, first collect a count of correctly clustered objects in each cluster and sum these counts together. Dividing this number by the total number of objects produces an accuracy value in the range [0,1]. Because it is not known which learned cluster corresponds to each true cluster, the accuracy calculation is computed for all permutations of clusters and the maximum score is returned.

3.1. The Impact of an Incorrect Hypothesis

This experiment uses artificial data that is generated according to the RPPM specification, using the model parameters described in Section 2.3.

As mentioned earlier, many researchers make the assumption that all relations have a pull tendency. Instead of using RPPM to infer the edge model parameters, we fix them to initial values in order to simulate the effect of clustering data when different prior assumptions about relation tendency are made. We tested the accuracy of clustering under several different modeling assumptions:

1. Ignoring relations,
2. Assuming all relations are push relations,
3. Assuming all relations are pull relations, and
4. Using the true lambda values for all relations.

The generated data sets have 250 two-dimensional objects in two clusters with three different relation types. The features of the objects are generated from multidimensional Gaussians that were specified such that a significant overlapping of the clusters existed. The first relation type is a pull-type relation, with $\lambda_I = 0.8$ and $\lambda_O = 0.2$; the second relation type is a push-type relation, with $\lambda_I = 0.3$ and $\lambda_O = 0.6$; and the third is neither push nor pull, with $\lambda_I = 0.3$ and $\lambda_O = 0.3$. The correct hypothesis assumes the parameters listed above. The all-pull hypothesis assumes that the parameters are $\{0.9, 0.1\}$ for all three graphs and the all-push hypothesis assumes that the parameters are $\{0.1, 0.9\}$ for all three graphs.

Table 1 shows the clustering accuracy for each of the above cases for three different runs. The most important observation is that a significant performance advantage is present only when the correct hypothesis is made. This means that making an incorrect assumption about the tendency of a relation can have such serious repercussions that it is better to ignore the relations, unless the assumption is changed. This finding supports Neville et al.'s [6] conclusions: as discussed in Section 4, they concluded that it is best to ignore relations if they do not agree with the features. With RPPMs, the relations no longer need to be ignored, but instead, their tendencies must either be correctly hypothesized or inferred from observed data, as we have done for our next set of experiments.

Assumption:	Correct	All Pull	All Push
Graph & Features	0.9576	0.6640	0.5696
Graph Only	0.9564	0.6676	0.5768
Features Only	0.6432	0.6432	0.6432

Table 1. The impact of a poor hypothesis. When the edge hypothesis is incorrect, the optimal solution is not the correct clustering.

3.2. The Internet Movie Database Data Set

The Internet Movie Data Base (IMDB)[1] is an online resource containing information on movies, actors, directors and producers, including links indicating various types of interrelations between the objects. It has become a canonical data set for evaluating relational clustering algorithms. The data set used here is a small subset of the entire database that contains 508 actors and 2756 undirected edges of a single type, where an edge between two actors means that they starred together in the same movie. There are eight binary features—has-award, active-in-90s, genre-drama, genre-comedy, experienced, is-male, high-hsxrate, and many-movies—with no missing values. The titles are all self-explanatory, except for high-hsxrate, which is a numeric ranking from the Hollywood stock exchange[2]. The actors with a hsxrate greater than fifty have the feature high-hsxrate set to one. One method for testing a clustering algorithm is to remove one discrete feature value and use it as a class label. The goal, then, is to see if the clustering algorithm can use the remaining features to find a clustering that corresponds to the class label. The first two experiments investigate the data in this way. The third experiment uses all eight feature values, and a qualitative analysis is performed. Values reported in these experiments are the maximum of five random restarts of the simulated annealing algorithm.

Experiment 1: Full Inference of All Parameters The first set of experiments uses the lambda matrix es-

[1]`http://www.imdb.com`
[2]`http://www.hsx.com/`

Feature	Full Inference			Features Only		
	Max	Avg	Std	Max	Avg	Std
has-award	0.75	0.72	13.0	0.78	0.68	42.5
active-in-90s	0.75	0.72	10.4	0.82	0.72	29.5
high-hsxrate	0.64	0.61	25.8	0.60	0.57	17.8

Table 2. Clustering accuracy for the IMDB data set. Each row specifies the results when the given feature is used as a class label.

timator (Section 2.4) to investigate whether the single relation type—starred-with—is a push-type, pull-type, or neither type of relation. We performed a series of trials, each using one of the eight features as a class label. In all trials, the lambda estimator found a push-type relation. The accuracy using RPPM with parameter estimation was not significantly different on average than clustering on the features only (Table 2).

Although RPPM did not improve the *average* accuracy, the *variance* of the trials using inference was lower. For example, when active-in-90s was used as a class label, the best features-only score was much higher, but the variance is also much higher, so that the average accuracy—0.72—is the same as the average accuracy using inference, which is also 0.72. The presence of both qualities implies that the consideration of relations guides the search through a particular portion of the search space, and that the relations are indeed generated by a specific clustering.

Experiment 2: Fixing the Lambda Matrix Instead of allowing inference of the lambda matrix at each iteration, these trials keep the parameter matrix fixed. Two different situations are analyzed, each one fixing Λ to first indicate a pull relation type, and then to indicate a push relation type. The values in Λ were chosen with guidance from the results in Experiment 1. There are two purposes for this experiment. The first is to validate the results of Experiment 1, that the lambda estimator is effective. The second is to see if assuming a different relation type does indeed harm performance, as claimed in Section 3.1.

As can be seen in Table 3, the use of a push-type lambda matrix with similar values to the one inferred in several trials in Experiment 1 produces similar results to Experiment 1. In fact, the average performance is identical. The second column shows what occurs when the relation tendencies are reversed: that is, when the inter-cluster lambda values are presumed to be low and the intra-cluster lambda values are presumed to be high. Just as with the artificial data sets, making an incorrect assumption harms performance.

Experiment 3: Full Unsupervised Learning As a final test of RPPM on the IMDB data set, we ran the

Feature	Fixed Pull Type			Fixed Push Type		
	Max	Avg	Std	Max	Avg	Std
has-award	0.74	0.71	8.0	0.63	0.55	26.0
active-in-90s	0.74	0.71	3.7	0.59	0.57	14.4
high-hsxrate	0.65	0.63	7.8	0.53	0.52	4.3

Table 3. Clustering accuracy for the IMDB data set when Λ is fixed.

Cluster 0	Shared	Cluster 1
genre-comedy	has-award	genre-drama
	active-in-90s	experienced
	is-male	high-hsxrate
		many-movies

Table 4. High-frequency features for both clusters.

algorithm over all features. The results of the previous two experiments suggest that the data fits relatively well to a two-cluster configuration, so this trial searches for a two-cluster set.

The analysis involves measuring the frequency that each feature has a value of 1 for each object in both clusters. These frequency counts help to characterize the actors in the group. For example, if one cluster has a high frequency of high-hsxrate and the other does not, we can infer that the first cluster contains more popular actors than the other one. Table 4 summarizes the results of the frequency counts. Cluster 0 had only one frequently positive-valued feature in genre-comedy. This is not to say that it is its only defining characteristic. It is also more frequent that the actors in Cluster 0 are not drama stars, experienced, or have a high hsx-rate, since cluster one had positive values for those features with a higher frequency. Just as in the previous experiments, the relation type inferred is a push-type relation. The most likely interpretation of an edge given these results is that experienced, popular actors tend to star with inexperienced, less popular actors. This is intuitive, since movies with many popular stars are rare compared to movies that have only one superstar.

4. Related Work

Neville and Jensen [6] were some of the first researchers to develop clustering algorithms that use *both* object features and relations between objects. Their method was to cluster relational data by incorporating object feature similarity into the relational graph as edge weights. Then adapted graph cutting algorithms could be used to cluster the data hierarchically. Their results showed a key concept in relational data clustering, which is that relational data clustering does well if both the edges and object features individually provide a rich source for finding a particular clustering. When this is the case, performance is high, but when the edges do not agree with the features, the performance is worse than if the edges were completely ignored. In the context of RPPM, since their algorithm searched for a minimum cut of a graph, it was essentially making a pull-type assumption for all relations. Their algorithm performs well when the relation is a pull-type, but does not perform well with push-type relations.

The most similar model to the RPPM is the Latent Group Model (LGM) [7]. LGMs have the same dependency for object features that RPPMs have: each feature value depends solely on the object's cluster membership. Their model differs from RPPMs in their notion of edge existence. They assume that in addition to objects belonging to clusters, objects also belong to a different, hidden grouping in the data, referred to as a latent group. These latent groups represent *coordinating objects* that are not part of the data set; these coordinating objects represent the reason that objects are in the same latent group. Edges then depend solely on the groups to which its endpoint objects belong. An object's cluster also depends on the group the object belongs in, so that objects that are in the same group are more likely to be in the same cluster. Since edges help determine the group an object belongs to, they indirectly affect the cluster membership of the object. LGMs differ from RPPMs in terms of their edge existence models. In RPPMs, edge existence depends on the cluster membership of the objects, but in LGMs, the cluster membership (indirectly) depends on edge existence.

Finally, Kubica et al. [5] present a model where links exist as the result of some event (e.g., a phone call or a meeting). They use this model to detect groups of associated people from a social network. Their model is effective in representing such a situation, but does not generalize well to other problems. Further work is necessary to determine if RPPM can detect the same groups as Kubica et al.'s work.

5. Future Work

Discrete Data: Many relational data sets in existence do not contain any numeric data. RPPM's flexibility for specifying the kinds of distributions for object and edge existence will allow the use of a discrete method, such as Latent Dirichlet Allocation [2], to develop a method for incorporating both relations and discrete features into a single model.

Heterogeneous Data: Feature-based comparison of heterogeneous objects is difficult and sometimes impossible. RPPM can be used to cluster data without comparing object features, as showed in our experiments. If a reasonable model for edge existence between different typed objects could be developed, RPPM could be used to cluster heterogeneous data.

6. Conclusion

The Relational Push-Pull Model is a unique framework that models the specific tendencies that a relation can have with regards to a specific clustering. This model expands the space of possible clusterings beyond previous works that considered all relations to be pull-type relations. By searching in this expanded space, we showed that better clusterings may be discovered for different data sets. However, the complexity of relational data sets and the computational complexity of using a probabilistic approach to relational data clustering are both major limitations of Relational Push-Pull Models. Given additional modifications to the model, further refinement of the search method, and the inclusion of discrete methods or subroutines, we believe that Relational Push-Pull Models will become a useful tool for clustering complex relational data.

References

[1] I. Bhattacharya and L. Getoor. Entity resolution in graph data. Technical Report CS-TR-4758, University of Maryland, October 2005.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993 – 1022, 2006.

[3] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3:679–707, 2002.

[4] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proc. ICML-02*, pages 259 – 266, 2002.

[5] J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *Proc. AAAI-02*, pages 798–804, 2002.

[6] J. Neville, M. Adler, and D. Jensen. Clustering relational data using attribute and link information. In *Proceedings of the Text Mining and Link Analysis Workshop*. IJCAI-03, 2003.

[7] J. Neville and D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proc. ICDM-06*, 2006.

[8] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proc. IJCAI-01*, pages 870–878, 2001.