12-2005

Scalable and Robust Group Discovery on Large Transactional Data

Patrick Pakyan Choi
Carnegie Mellon University

Andrew W. Moore
Carnegie Mellon University, awm@cs.cmu.edu

Jeremy Kubica
Carnegie Mellon University

Scalable and robust group discovery on large transactional data

Patrick Pakyan Choi Andrew Moore Jeremy Kubica

CMU-RI-TR-05-60

December 2005

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

The need for time-critical analysis and understanding of the underlying group structure from transactional data has been growing in domains such as law enforcement and customs. Kubica et al. (2003) proposed k-groups, an algorithm based on probabilistic generative model for discovering underlying groups in data. Even though k-groups is reported to be signficantly faster than its predecessor GDA (Kubica et al., 2002), k-groups is too slow and memory-intensive for large data in practice. This paper presents XGDA, a framework for scalable and robust group discovery. Evaluation of the performances of XGDA and k-groups shows that XGDA can handle extremely large datasets in reasonable time and yields more robust solutions than k-groups.

Contents

1 Introduction

Having automated tools to help us analyze vast amount of data is crucial and beneficial. Data is worthless unless it is analyzed and understood. However, we live in an age of information overflow. Large-scale data analysis is difficult and time-consuming, if not impossible, for humans.

Learning and understanding the underlying group structures from collected transactional data is an important task in domains such as law enforcement and customs. These domains not only require the automated group discovery system to yield accurate solution, but also to yield the solution fast. Therefore, a group detection algorithm has to be scalable and accurate in applications where time-criticality is of utmost importance.

XGDA is a group discovery system developed in response to the growing need of time-critical analysis and understanding of large transactional data. XGDA not only can yield hundred-fold speedup over our previous group detection algorithms, GDA (Kubica et al., 2002) and k-groups (Kubica et al., 2003). XGDA can also produce a more robust analysis than other algorithms through its model selection mechanism. XGDA also produce a hierarchy of the discovered groups to provide a intuitive visualization and understanding of the data for human analysts.

2 Overview of XGDA

A group is a collection of entities sharing a common identity or charateristics. For example, a family is a group of individuals having intimate kin-based relationship. A religious institution is a group of people with a common religion. People may tend to hang out together more tend with those in the same group, either due to necessity or preference. Therefore if we observe that some people met up consistently, we may infer that these people come from the same group. Each person may belong to multiple groups, and groups may have overlapping membership. For example, a student may be taking several courses, and these courses have the said student enrolled.

Applying the intuition of frequent gathering among ingroup members, group detection algorithms can discover groups of entities in transactional data. Transactional data is records of entities co-occuring. Publication data is such an example, where each record is a paper published, entities in the record are co-authors, and groups may represent research groups or researchers with similar research interests. The notion of transactional data can go beyond interactions of human beings. For example, a record in supermarket shopping data is a shopping basket rung up at cash register, entities in the record are the commodities in the shopping basket, and groups may indicate sets of complementary goods.

GDA (Kubica et al., 2002) and k-groups (Kubica et al., 2003), our previous group detection algorithms, received very positive feedback in their uses in several government agencies. GDA and k-groups are based on a probabilistic generative model and produce coherent groups that are consistent with prior knowledge. K-groups produces a coarser analysis than GDA in much shorter turnaround time through its use of a heuristic similar to K-means. However, even though k-groups achieves significant speedup

over GDA, k-groups is impractical for time-critical large-scale group discovery application. In our evaluation, k-groups failed to yield results within seven days for CiteSeer dataset (304490 entities and 385923 records). Besides its poor scalability, k-groups has another shortcoming. As any other data-mining algorithms, k-groups has its own model assumptions. Beliving that one set of model assumptions will be met in the wide variety of domains is unrealistic. Therefore in practice, k-groups may outperform other group detection algorithms on some data, and underperforms on other data.

XGDA is a system developed to address the shortcomings of k-groups in scalability and robustness. XGDA's hybrid approach to group discovery is capable of satisfying needs in a wide spectrum of application. For time-critical applications, XGDA's fast component can produce a rough analysis of a huge amount of data within minutes. For non-time-critical applications, XGDA's slow component can yield a robust group model through its model selection mechanism. Either step will produce a hierarchy of discovered groups to aid human understanding of the group structures.

3 Hybrid Approach to Group Discovery

The XGDA algorithm consists of two components. Both components analyze the data to find its underlying groups and a hierarchy of the groups. The fast component aims to provide an initial analysis of the data in a short amount of time. If time allows, the slower component will be run to obtain more detailed analysis.

3.1 Fast Component

The fast component analyzes the groups using recursive partitioning algorithm, which is a divide-and-conquer group detection algorithm. Each instance of the algorithm analyzes the interconnectivity of the given set of entities to create two disjoint subsets of entities that co-occur frequently within subset and co-occur infrequetly across subsets. The recursive partitioning algorithm starts with the entire set of entities, partitions the set into two, and recurses on each of the two partitions until a subset of entities is determined to be a tight-knited group. At the end of recursions, the algorithm outputs all the leaf tight-knited groups and also the tree of partitions as the hierarchy of the groups. Details of the recursive partitoning algorithm are described in Section 4.1.

3.2 Slow Component

The slow component can produce a higher quality group model at the cost of running time. The strategy of the slow component is based on the two intuitions: there is no single algorithm that is universally best, and the quality of models generated by some algorithms tend to improve less as they run longer. Therefore the strategy of the slow component for group discovery is to run multiple group detection algorithms for a short period of time and then choose the best group model for output. In our evaluation, we showed that this strategy of splitting time among multiple algorithms often produces model that is more accurate than model generated by running only a single algorithm.

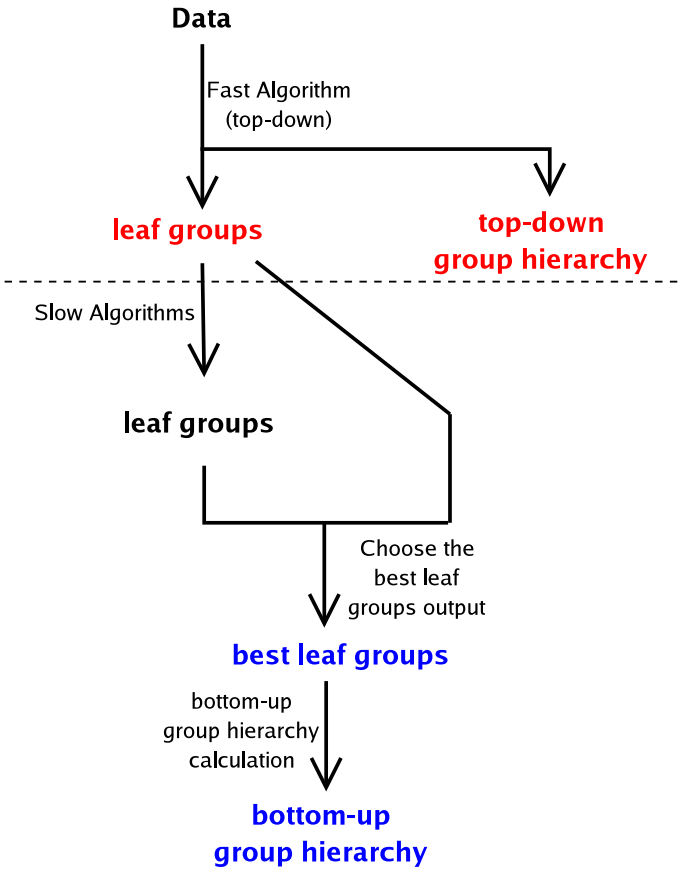

Figure 1: Flowchart of the XGDA algorithm. The fast component, which is the part above the dotted line, provides an coarse analysis of the data. The part below the dotted line is the slow component and can produce better results. The slow component will be run only if the fast component finish within the time limit given by the users. The items in red are the output from the fast component. The items in blue are the output from the slower component.

In our implementation, the slow component may run up to three additional group detection algorithms: k-groups initialized randomly, k-groups initialized by the group model of recursive partitioning algorithm, and another spectral clustering algorithm (Ng et al., 2001). Incorporating new algorithms to the slow component is easy, therefore the XGDA framework is extendable. To choose the best model of groups, we use an heuristic of evaluating the capability of the model to predict future co-occurrence of entities, and evaluate each model on the data. The model with best predictive capability will be chosen as the output of XGDA. The model selection heuristic is described in Section 5. XGDA also generates a hierarchy of the chosen group model by an algorithm described

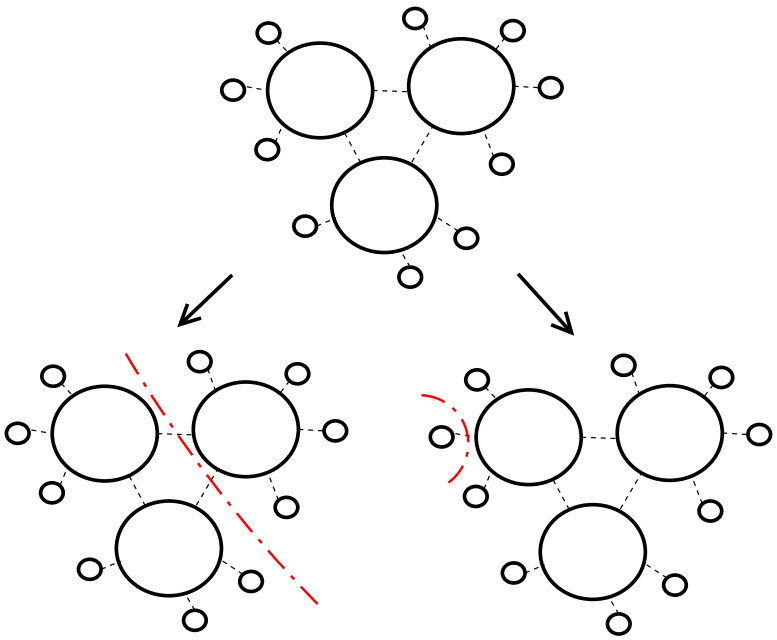

Figure 2: We demonstrate the possible scenarios of running the fast recursive partitioning algorithm on a data set that contains some tightly-connected groupings and a lot of flotsam. While the result varies with each data set, the partitioning algorithm favors finding a balanced split where the two daughter groups have roughly equal sizes (left scenario) than singling out small floatsam (right scenario). Therefore, the group hierarchy tree constructed by the fast algorithm is usually relatively balanced. And the floatsams are usually found as leaf groups deep down in the tree.

in Section 6.

4 Group Detection Algorithms

We describe the group detection algorithms used in the XGDA. Each algorithm has its own model assumptions and charaterics. Recursive partitioning is a spectral clustering-based algorithm which recursively bipartitions the set of entities according to a eigenvector of the adjacency matrix of the co-occurence data. K-groups is an algorithm based on probabilistic generative model and uses hill climbing heuristic for optimization. Ng-Jordan-Weiss algorithm is another spectral clustering-based algorithm which partitions the set of entities by running k-means algorithm on the subspace spanned by multiple eigenvector of the adjacency matrix.

Throughout this paper we use R and N to denote the number of records and the number of entities in the co-occurrence data, respectively.

4.1 Recursive Partitioning

Recursive partitioning algorithm determines the composition of the groups through analyzing the eigenvalues and eigenvectors of the adjacency matrix.

In spectral graph theory, it is well known that the Fielder vector, which is the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix of a graph, contains valuable information for partitioning the graph (Chung, 1997).

Intuitively, we first transform the co-occurrence data into a graph. Each entity is turned into a vertex. If the data indicates two entities co-occurred, we connect the two corresponding vertexes in the graph with an edge, and set the weight of this edge to be the number of meetings between this two entities in the data. We can think of this graph as a network and the thickness of the pipe between any two nodes increases with the strength of association of the two nodes. Then, we partition the graph into two by finding a cut that minimize the loss of the flow in this network while maintaining a balance in sizes of the two partitions. This cut can be found by using the Fielder vector. Once we cut the graph into two partitions, we continue to find such cut in each of the two partitions. We keep on partitioning the graph into finer and finer partitions recursively and stop if the vertices in a partition are strongly connected. Through this recursive partitioning procedure, we get all the fine partitions in the graph. Finally we transform the partitions into groups: all the vertices in a partition are treated to be members in the same group, and the number of partitions in the group indicates the number of groups in the data.

This recursive spectral clustering algorithm has multiple advantages. Firstly, it is capable of finding the groups without prior knowledge of the number of groups. Secondly, it is much faster than full-blown spectral clustering algorithms, and thus can very quickly produce an crude analysis of the underlying groups even for large datasets. Thirdly, the recursive nature of the algorithm provides a natural hierarchy of the groups.

On the other hand, there are two disadvantages of the recursive partitioning algorithm. This algorithm may not be as accurate as other spectral partitioning methods that use multiple eigenvectors to find the cut because information loss in discarding eigenvectors. Also, this algorithm cannot model overlapping groups. However, the potential detrimental effects on the accuracy of the groups found is mitigated by the model selection in the slow component.

4.1.1 Derivation

He et al. (2002) showed that the optimal partition according to the normalized cut criterion can be obtained by solving for $\vec{x}_{soln}$ the eigenvector associated with second smallest eigenvalue of the general eigensystem

$$(\mathbf{D} - \mathbf{W})\vec{x} = \lambda \mathbf{D}\vec{x} \tag{1}$$

where $\mathbf{W}$ is the the co-occurrence data's weighted adjacency matrix, which is a $N \times N$ symmetric matrix where the entries (i, j) and (j, i) is the number of times entity i and entity j co-occur in the data, and the diagonal (i, i) defined to be all zeros,

$\mathbf{D}$ is the weighted degree matrix, which is a diagonal matrix with diagonal entries $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$, and $\vec{e} = [1 \ldots 1]^\mathsf{T}$.

The solution of the second smallest eigenpair of the eigensystem (1) is related to the solution of the second largest eigenpair of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ (See Appendix). Denote the eigenvector of this eigenpair as $\vec{x}_{trans}$. It can be shown that $\vec{x}_{soln} = \mathbf{D}^{-1/2}\vec{x}_{trans}$.

Since all the eigenvalues of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are at most 1, we know that $\mathbf{D}^{1/2}\vec{e}$ is the eigenvector associated with the largest eigenvalue of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. Therefore, $\vec{x}_{trans}$ can be solved as the eigenvector associated with the largest eigenvalue of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - \frac{\mathbf{D}^{1/2}\vec{e}\vec{e}^\mathsf{T}\mathbf{D}^{1/2}}{\vec{e}^\mathsf{T}\mathbf{D}\vec{e}}$. We can apply the power method (Kincaid and Cheney, 1991) for this computation.

After the power method, we can compute the $N \times 1$ $\vec{x}_{soln}$ through normalizing $\vec{x}_{trans}$ by $\mathbf{D}^{-1/2}$. And we threshold on $\vec{x}_{soln}$ to find the partition solution (Shi and Malik, 1997). We check the corresponding value in $\vec{x}_{soln}$ for each vertex, and we put it into one partition if the corresponding value is positive, and another partition if otherwise.

4.1.2 Computational Optimization

Computing an eigenvector by the power method is a $O(N^2)$ operation, and is intractable on large dataset with millions of entities. We improve the time complexity to $O(|\mathbf{W}| + N)$, where $|\cdot|$ is the number of non-zero entries in the matrix, by representing $\mathbf{W}$ in a sparse format.

Our goal is to computing $\vec{x}_{trans}$ the eigenvector associated with the largest eigenvalue from $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - \frac{\mathbf{D}^{1/2}\vec{e}\vec{e}^\mathsf{T}\mathbf{D}^{1/2}}{\vec{e}^\mathsf{T}\mathbf{D}\vec{e}}$. The original power method is

Initialize $\vec{x}_{trans}$ to $[1 \ldots 1]^\mathsf{T}$.

Iterate

$$\vec{z} := (\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} - \frac{\mathbf{D}^{1/2}\vec{e}\vec{e}^\mathsf{T}\mathbf{D}^{1/2}}{\vec{e}^\mathsf{T}\mathbf{D}\vec{e}})\vec{x}_{trans}$$
$$\vec{x}_{trans} := \frac{\vec{z}}{\|\vec{z}\|}$$

until convergence.

Let us denote $\mathbf{B}$ as $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, C as $\vec{e}^\mathsf{T}\mathbf{D}\vec{e}$, and $\vec{q}$ as $\frac{\mathbf{D}^{1/2}}{C^{1/2}}$. Then, the iteration of the original power method becomes

$$\begin{aligned}\vec{z} &:= (\mathbf{B} - \vec{q}\vec{q}^\mathsf{T})\vec{x}_{trans} \\ &= \mathbf{B}\vec{x}_{trans} - \vec{q}\vec{q}^\mathsf{T}\vec{x}_{trans} \\ \vec{x}_{trans} &:= \frac{\vec{z}}{\|\vec{z}\|}\end{aligned}$$

Computing $\mathbf{B}$ and $\mathbf{B}\vec{x}_{trans}$ are both $O(|\mathbf{W}|+N)$. And we can compute $\vec{q}\vec{q}^\mathsf{T}\vec{x}_{trans}$ in $O(N)$ time:

Let $\vec{z^*}$ be $\vec{q}\vec{q}^\mathsf{T}\vec{x}_{trans}$, and $\mathbf{Q}$ be $\vec{q}\vec{q}^\mathsf{T}$, that is, $\mathbf{Q}_{ij} = \vec{q}_i\vec{q}_j$.

For $i = 0$ to $n - 1$,

$$\begin{aligned}\vec{z^*}_i &:= \sum_{j=0}^{N} \mathbf{Q}_{ij}\vec{x}_{trans\,j} \\ &= \sum_{j=0}^{N} \vec{q}_i\vec{q}_j\vec{x}_{trans\,j} \\ &= \vec{q}_i \sum_{j=0}^{N} \vec{q}_j\vec{x}_{trans\,j}\end{aligned}$$

By precomputing $\sum_{j=0}^{N} \vec{q}_j\vec{x}_{trans\,j}$ (an $O(N)$ operation), we can compute each $\vec{z^*}_i$ in $O(1)$ time and thus the entire $\vec{z^*}$ vector in $O(N)$ time. Therefore the sparse power method has $O(|\mathbf{W}| + N)$ time complexity.

4.2 K-groups

The k-groups algorithm (Kubica et al., 2003) is a probabilistic generative model that assumes that links are the realization of the underlying groups. The algorithm optimizes the likelihood of the given transactional data in a k-means fashion: for each link in the data, it finds the group that maximizes the likelihood, and then updates the group memberships given the link. This process is repeated until the likelihood converges.

The advantage of the k-groups algorithm over the recursive partitioning algorithm is that k-groups can represent overlapping groups in data.

The k-groups algorithm cannot determine the number of groups in data automatically, and thus requires it as input paramter. Therefore XGDA runs the recursive partitioning algorithm, count the number of groups found, and supply that number k-groups algorithm.

K-groups is an memory extensive algorithm. Therefore XGDA do not run the k-groups algorithm if the recursive partitioning algorithm found more than 200 groups.

4.3 Recursive Partitioning + k-groups

The hill climbing optimization heuritic of k-groups can result in a solution in a local maxima. Therefore the initialization of k-groups may affect the quality of the solution generated by k-groups. By default, k-groups algorithm initializes randomly for optimization. Alternatively we can specify the group model generated by recursive partitioning algorithms to be used for initialization of k-groups. This procedure may improve the quality of the model produced by k-groups, as the group model generated by recursive partitioning algorithm may be more reasonable than that generated randomly.

Due to the k-group memory allocation constraint, this algorithm will not be executed if more than 200 groups are found by recursive partitioning.

4.4 Ng-Jordan-Weiss Spectral Clustering

Ng et al. (2001) proposed an algorithm for partitioning a graph using multiple eigenvectors simultaneously. The algorithm computes k largest eigenvectors of $D^{-1/2}WD^{-1/2}$, the normalized Laplacian adjacency matrix of the co-occurrence data. In our implementation, we keep all eigenvectors with eigenvalues larger than 0.3 of the largest eigenvalue. These k selected eigenvectors are then normalized to unit length. K-means is then run on the subspace spanned by these normalized eigenvectors to cluster the entities into predetermined number of disjoint groups.

This algorithm requires the number of (groups) as input parameter. As in the case of k-groups, we use the number of groups found by recursive partitioning algorithm as input. The Ng-Jordan-Weiss algorithm is intractable for dataset with many entities. Therefore, we do not run this algorithm if the dataset has more than 500 entities.

5 Model Selection

Because group detection algorithms have different assumptions and parameters, there may not be a universally best algorithm. Therefore, instead of running a single algorithm for a long period of time, XGDA runs multiple algorithms for a shorter period of time, and then tries to choose the best model from the generated models from these algorithms.

Model selection is a topic rich in research and theory. In our design of XGDA, we had to sacrifice accuracy for speed, as XGDA is primarily designed to be scalable and to operate in time-critical domains. Therefore, we adopted a simple model selection heuristic, and implemented some tricks and data structures to speed up the computation.

In our evaluation of learned model, we apply the following definition. The ground truth label of a pair of entities is positive if the pair of entities co-occurs in the data, and is negative if the pair that does not co-occur in the data. A pair of entities is labelled as predicted positive if at least one group in the learned model contains both entities, and the pair is predicted as negative if none of the groups in the model contains both entities. With these definition, we can compute the true positive rates and false positive rates of the models on the data (refer to Table 1.) Note that the true positive rate and false positive rate computed is a optimistic estimation of the acutal prediction rates on unseen data, because this implementation uses the same data for both learning and evaluating the model, and thus may be prone to overfitting.

Therefore, the above definition turns each learned model into a classifier. And the true positive rate (tpr) and false positive rate (fpr) of the model on the data can be calculated by

$$tpr = \frac{TP}{TP + FN}$$
$$fpr = \frac{FP}{FP + TN}$$

where TP is the number of the true positives, FN is the number of the false negatives, FP is the number of the false positives, and TN is the number of true negatives.

		Group Model	
		share at least one group	share no group
Data	co-occur	True Positive	False Negative
	did not co-occur	False Positive	True Negative

Table 1: Confusion matrix for evaluating a group model.

The scoring metric for comparing various model would be one that is related to the area under Receiver Operating Characteristic curve. Receiver Operating Characteristic (ROC) curve is a popular way of summarizing and visualizing the performance of a two-class classifier. ROC curve is a plot of the true positive rate of a classifier against its false positive rate. A ROC curve that is close to the upper-left corner indicates good predictive performance.

The area under ROC curve (AUC) of a classifier is the area under the ROC curve of the classifier. Higher AUC indicates better predictive performance under the assumption of lack of information on the class distribution and misclassification costs.

To evaluate the AUC of a group model, we have to generate its ROC curve. We compute the tpr and fpr of the model on the data, and treat this pair as a single point in the ROC space. The point (fpr, tpr) together with $(0, 0)$ and $(1, 1)$ forms the ROC curve of the model, and thus the area under the curve can be computed. In the example of Table 3, the model has a true positive rate of 0.8, and a false positive rate of 0.3. The bold black line is the ROC curve, and the purple area is the area under the curve.

The scoring function for the model selection in XGDA is defined as the difference between tpr and fpr, which is proportional to the AUC of the model on the data.

$$\begin{aligned} S(\text{Model}, \text{Data}) = S(tpr, fpr) &\equiv tpr - fpr \\ &\propto \frac{tpr\ fpr}{2} + \frac{(1 + tpr)(1 - fpr)}{2} \\ &\equiv \text{AUC} \end{aligned}$$

With this scoring function, XGDA select the model that has the best score. That is, $\text{Model}_{\text{XGDA}} = \text{argmax}_{\text{model}_i \in \text{ model list}}\, S(\text{model}_i, Data)$

The reason of defining the scoring function to be related to AUC is that this measure

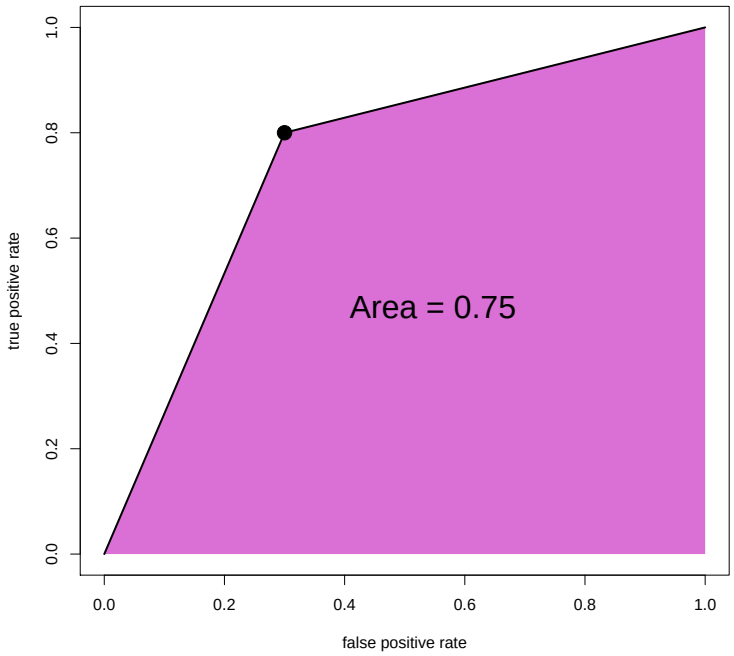

Figure 3: Example of a ROC plot. The classifier has a false positive rate of 0.3 and a true positive rate of 0.8. The bold line connecting $(0, 0)$, $(0.3, 0.8)$, and $(1, 1)$ is the ROC curve. The purple shaded indicates the area under the ROC curve. And the area can be calcaulted to be 0.75.

is not affected by the class inbalance. As a contrast, classification accuracy, a popular evaluation criteria, is sensitive to the class porpotion in the data. The further imbalance away from equal number of positive and negative instances, the worse the accuracy is capturing the actual prediction ability of the classifier. And very often, in transaction data there is a usually abundance of negative class compared to positive class. The reason being, if there is underlying groups in the data, then each entity should only be connected with a limited number of other entities (those that shares the same group, and possibly some noise links). Therefore, the insensitivity to class balance of AUC makes it a suitable choice of scoring function. In fact, using AUC as scoring function implicitly assumes that we have no prior information of the class distribution and the misclassification cost.

Depending on the size of the data and the user alloted time limit, up to four algorithms, recursive partitioning (RP), k-groups with random initialization (k-groups), k-groups initialized by the groups found by recursive partitioning (RP + k-groups) and Ng-Jordan-Weiss spectral clustering (NJW), will be run in our implementation of XGDA, and the XGDA output model will be selected from them. This can be easily

extended to incorporate other group detection algorithms.

5.1 Implementation

From the data, we construct the sparse adjacency matrix. From the group detection algorithms, we obtain a sparse group model in the form of a list of lists: each entry of the outer list represents a group; the inner list is a sorted list of entity IDs that keeps track of all the entities belonging to that group.

To speed up the calculation of true positive and false negative counts, we construct a inverse model from the derived group model. The inverse model is also a list of list: each entry of the outer list represents an entity, and the inner list is a sorted list of group IDs in which that this entity belongs to.

To compute the confusion matrix, we first build a hashtable from the group model to store all the connecting pairs of entities in the model. The group model is iterated to extract all connected pairs of entities to be inserted into the hashtable, using the two sorted entity IDs as key. Then we efficiently iterate all pairs of co-occurrring entities from the sparse adjacency matrix, and for each pair, we determine whether it is a true positive or a false negative by querying the hashtable. It is a false negative if the pair is in the hashtable, that is, connected in the group model. And it is a true positive if otherwise.

We use the fact that the number of elements in the hashtable $|G|$ is the number of unique pairs of connected entities in the group model, and the sum of true positives and false positives is equal to the number of of unique pairs of connected entities in the group model by definition, to calculate the false positive count. The false positive count is calculated by subtracting the count of true positives from the number of elements in the hashtable.

We also know that there are $\frac{N(N-1)}{2}$ possible pairs of entities, and true positives, false positives, true negatives, and false negatives add up to $\frac{N(N-1)}{2}$. Therefore, we can compute the count of true negatives by subtraction.

1. count TP
2. count FN
3. $FP = |G| - TP - FN$
4. $TN = \frac{N(N-1)}{2} - TP - FN - FP$

The time complexity of building the hashtable is $O(\#G \times |\text{Largest group}|^2)$, where $\#G$ is the number of groups in the model, and $|\text{Largest group}|$ is the number of entities in the largest groups in the model. The space complexity of the hashtable is $|G|$, the number of unique pairs of connected entities in the model. Empirically, we found that for huge datasets, this evaluation implementation slows down significantly. As the size of dataset increases, the structure in data becomes more complex and thus leading to more groups found. And also it is more likely to have a very large group due to the behavior of the algorithms. These factors can contribute the significant increase in the computation time. We also found that the hashtable may use up a

lot of memory for huge datasets, as the number of entries in the hashtable is at least $\frac{|\text{Largest group}|(|\text{Largest group}|-1)}{2}$, which can become prohibitively large for a bad-behaving group model.

5.2 Alternative formulation

We describe here an alternative decision-theoretic method for selecting the best group model from all algorithms (and paramters).

We build up a list of model incrementally by the ROC Convex Hull Method (Provost and Fawcett, 1997). We iteratively run all combinations of algorithms and number of groups, and keep the output group model only if the test-set true positive rate and false positive rate represent a point in the ROC space that is outside the convex hull of the current list of group models. At the end of this incremental operation, we are left with a compact list of group models.

We then construct a cost curve (Drummond and Holte, 2000) from these group models. A cost curve is a plot of normalized expected cost versus probability-cost, which is defined as

$$PC(+) = \frac{p(+)C(-|+)}{p(+)C(-|+) + p(-)C(+|-)}$$

where $p(+)$ and $p(-)$ are the probabilities of the entitiy-pair co-occurring and entitiy-pair not co-occurring, respectively, and $C(-|+)$ and $C(+|-)$ are the costs of a false negative and a false positive, respectively.

The normalized expected cost is a function of probability cost, and is defined to be

$$\begin{aligned} NEC &= (1 - tpr)\ PC(+) + fpr\ PC(-) \\ &= (1 - tpr - fpr)\ PC(+) + fpr \end{aligned}$$

We can see that each (tpr, fpr) point in the ROC space corresponds to a line with intercept tpr and slope $1 - tpr - fpr$ in the cost space. By drawing all the operating lines of the learned models in the cost space, we can find the minimum expected cost model for any particular operating ranges of probability cost.

Let us visualize the processing of model selection from the example of Figure 4. In this example, we have three group models, red, green, and blue. And the red, green, and blue models achieve true positive rates and false positive rates of $(0.05, 0.4)$, $(0.2, 0.8)$, and $(0.35, 0.9)$ respectively. In the cost space, we plot the cost line for each model. We also plot the costs of choosing two trivial models: one that has only one group containing every entities, and one that has N disjoint groups each containing one entity. These two models are represented by the two diagonals in the cost space. Suppose the process of data generation is a representative sampling of the population and the events, we can approxmiate the probability of the entities co-occurrence with the empirical value computed from the data. For example, the data suggests that the probability of pair of entities co-occurring is 0.2, and we assume equal cost for false positives and false negatives. Then we can calculate the probability-cost to be 0.2, and see from the ensemble cost curve that the red model is the best ones under our assumptions.

If our knowledge of the relative misclassifying cost is a distribution, we can compute the distribution of the probability-cost, and then find the expected cost of each model by integrate its cost curve with the distribution over the probability-cost. Again, we choose the model that minimizes the expected cost. This can be extended to the scenario where we know only the range of the relative cost. In that scenario, we can assume a uniform distribution over the specified range, and proceed accordingly.

This decision-theoretic method chooses the best model from a joint space of algorithm and number of groups, as opposed to the PWE method that chooses the best model from candidates that are generated with the number-of-groups parameter predetermined by a hueristic.

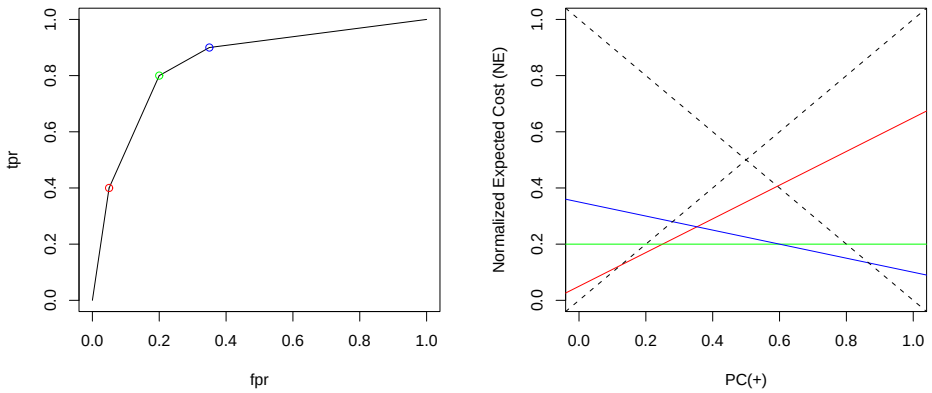

Figure 4: An example of building ensemble group model. The left plot shows a ROC curve of three group models (denoted as the red, green, and blue point in the plot). The right plot is the dual representation of the group models in the cost curve space. The two dotted diagonal lines represents the cost of a group model with a single group containing all entities, and the cost of a group model with N disjoint groups each containing one entity. Depending on the range of the probability-cost of the problem, we select the model that minimize the expected cost over that range. For example, if the experts claim that the co-occurrence data should have a positive prevalance of 0.2, and a false positive is as serious as a false negative, then the probability-cost is 0.2. And therefore, we should choose the red model as it has the lowest expected cost among all models at probability-cost of 0.2.

6 Constructing Group Hierarchy

A hierarchy of discovered group can help us understand the interactions of the entities and groups. The hierarchy may provide us intuitions on how the groups relate to each other.

Let us define Pairwise Error (PWE)

$$\text{PWE} \equiv \sum_{g=1}^{\#\text{ Groups}} \#\text{ pairs of entities in group } g \text{ that has no evidence of co-occurrence in the data}$$

We construct a group hierarchy in a bottom-up fashion. We find the two groups that will produce a merge that results in minimum PWE, merge these two groups, and then repeat the search until there is only one single group remaining.

Each iteration of finding the pair of minimum PWE merge can be formulated as

$$\underset{i,j \in Groups}{\operatorname{argmin}} \; PWE_{i \cup j}$$

because merging two groups has no effect on the PWE of any other group by definition.

Let us consider the simple scenario where there are only two overlapping groups of entities, group$_i$ and group$_j$. As the Venn's diagram in Figure 5 shown, the entities can be broken down into three disjoint sets: $s_{i \setminus j}$, those who belong to group$_i$ but not group$_j$; $s_{j \setminus i}$, those who belong to group$_j$ but not group$_i$; $s_{i \cap j}$, those who belong to the both groups.

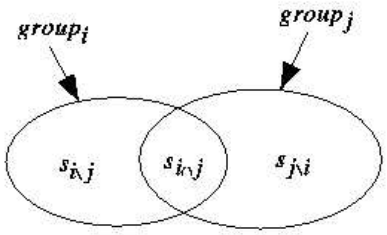

Figure 5: A Venn's diagram showing the three sets of members composing group$_i$ and group$_j$. $s_{i \setminus j}$, those who belong to group$_i$ but not group$_j$; $s_{j \setminus i}$, those who belong to group$_j$ but not group$_i$; $s_{i \cap j}$, those who belong to the both groups.

Merging group$_i$ and group$_j$ may introduce new connections between pairs of entities. New connection may be formed by linking one entity in $s_{i \setminus j}$ and one entity in $s_{j \setminus i}$. Since the unsupported links within set $s_{i \cap j}$ will be double counted by merging the two groups, we should subtract it from the new count. Therefore $PWE_{i \cup j}$, the new pairwise error of the merged group can be calculated by

$$PWE_{i \cup j} = PWE_i + PWE_j + \tilde{c}(s_{i \setminus j}, s_{j \setminus i}) - \tilde{c}(s_{i \cap j}, s_{i \cap j})$$

where $\tilde{c}(a, b)$ is the count of pairs of which an entity in set a is connected in the group model but does not co-occur in the data with an entity in set b.

We exploit two facts to speedup the computation. Note that the number of pairs of entities supported by data and the number of pairs unsupported by data must sum to

the number of possible pairs of entities in the set. That is, $\tilde{c}(a,b) + c(a,b) = N_a N_b$, where N_a and N_b are the number of entities in the two disjoint sets a and b respectively. Coupling with the fact that $\tilde{c}(a,a) + c(a,a) = \begin{pmatrix} N_a \\ 2 \end{pmatrix}$, we can compute $PWE_{i\cup j}$ by an alternative formula,

$$PWE_{i\cup j} = PWE_i + PWE_j + N_{s_{i\setminus j}} N_{s_{j\setminus i}} - \begin{pmatrix} N_{s_{i\cap j}} \\ 2 \end{pmatrix} + c(s_{i\cap j}, s_{i\cap j}) - c(s_{i\setminus j}, s_{j\setminus i})$$

This formula allows the calculation of the PWE of the merged group without actually merging the two groups. The fast computation of $PWE_{i\cup j}$ makes searching of the best merge tractable.

6.1 Implementation

To calculate the PWE resulted by merging group$_i$ and group$_j$, we need to compute the two counts $c(s_{i\cap j}, s_{i\cap j})$ and $c(s_{i\setminus j}, s_{j\setminus i})$.

A simple algorithm of simultaneously traversing both sorted entityID arrays of group$_i$ and group$_j$ can yield all three sets of entities $s_{i\cap j}$, $c(s_{i\setminus j}$, and $s_{j\setminus i})$.

To find the count of supported links For each entity e in $s_{i\cap j}$, we efficently retrieve the list of entities co-occured with e in constant time from the sparse adjacency matrix in constant time. Then for each entity in the retrieved list, check whether it is in the set $s_{i\cap j}$ either through performing binary search on sorted entityID array of set $s_{i\cap j}$ or querying a hashtable populated with the entityID of set $s_{i\cap j}$. If this entity is in the set $s_{i\cap j}$, increment the count of $c(s_{i\cap j}, s_{i\cap j})$.

The count $c(s_{i\setminus j}, s_{j\setminus i})$ can be computed in a similar fashion. Retrieve lists of entities co-occured with an entity in $s_{i\setminus j}$, and check for each entity in those retrieved lists its existence in the set $s_{j\setminus i}$.

7 Experiments

We evaluate the performance of an algorithm using the cross-validated estimate of the AUC of the model generated by the algorithm. We randomly split the records in the data into ten sets. In each run, we combined eight sets to form the training set, and the remaining two sets to form the test set. We then run XGDA on the training set, and evaluate the group models on the testing set.

In our evaluation of learned model, we apply the same definition of positive and negative labels as in Section 5. The only difference is that instead of evaluating on the training data, here we evaluate the prediction of the group model on the testing data.

By varying the number of groups in our experiments, a group detecion algorithm can output a series of true positive rates and false positive rates. With this series of points in the ROC space, we can plot the ROC curve of a group detection algorithm. By repeating the evaluation with cross-validation, we obtain the average ROC curve by threshold averaging (Fawcett, 2003). Threshold averaging is averaging the ROC points according to the generating parameter.

7.1 Datasets

We evaluated XGDA on 13 datasets, 5 of which are synthetic data and 8 of which are real data. The datasets are collected from a wide variety of domains, such as academic publications, US customs, theater production, and sports teams.

7.1.1 Synthetic Data

1. *disjoint*
   2000 records, 500 entities. 10 disjoint groups. All entities are members of exactly one group. A few noise links.

2. *EAGLE-27*
   1926009 records, 9983 entities. The dataset is one of the EAGLE data.

3. *EAGLE-45*
   162639 records, 388474 entities. The dataset is one of the EAGLE data.

4. *EAGLE-63*
   103675 records, 260467 entities. The dataset is one of the EAGLE data.

5. *EAGLE-71*
   2149464 records, 5508508 entities. The dataset is one of the EAGLE data.

7.1.2 Real Data

1. *AUTON*
   95 records, 120 entities. The dataset describes co-author relationships among Auton Lab professors, students, and friends.

2. *CiteSeer*
   80000 records, 80106 entities. The dataset is a subset of the CiteSeer database, which records the academic publication information in areas related to Computer Science.

3. *Containers*
   344658 records, 9505 entities. The dataset records a subset of import shippments, and the entities are the shipping companies, importing firms, and exporting firms.

4. *IOBDB*
   3686 records, 29446 entities. The dataset is generated from the Internet Off-Broadway Database, and describes the relationships among cast members, crews, and production organizations through the show productions.

5. *MLB*
   2471 records, 5000 entities. The dataset is a subset of a larger dataset that describes the Major League Baseball player-team relationships.

6. *Robotics*
   1474 records, 1095 entities. This dataset describes co-authorship of papers, common interests, and/or student/advisor relationships in Carnegie Mellon's Robotics Institute.

7. *RSC*
   3145 records, 9500 entities. This dataset describes production and cast informations about the shows from Royal Shakespeare Company.

8. *Terrorist*
   5581 records, 4065 entities. This dataset describes links involving terrorist incidents, people, places and/or events. The data was manually gathered from the Web in 2003.

7.2 Results

Figure 6, 7 and 8 show the ROC curves of the group detection algorithms on the datasets. These plots show that XGDA produced the best model on most datasets. Table 2 summerizes the performances in AUC. On all datasets evaluated, XGDA outperforms the alternative strategy of running k-groups alone. These results provide support to our claim that more robust group model can be generated by running multiple algorithms for short amount of time than by running a single algorithm for a long time.

	RP	k-groups 150s	RP + k-groups 150s	NJW	XGDA	k-groups 375s
disjoint	**0.8624**	0.7276	**0.8628**	0.5954	**0.8628**	0.7816
EAGLE-27	**0.5120**				**0.5120**	
EAGLE-45	**0.5641**				**0.5641**	
EAGLE-63	**0.5433**				**0.5433**	
EAGLE-71	**0.5028**				**0.5028**	
AUTON	**0.7881**	0.6256	0.6259	0.5939	0.6259	0.6249
CiteSeer	**0.8098**				**0.8098**	
Containers	**0.6806**	0.5164	0.6039		**0.6806**	0.6332
IOBDB	**0.6351**				**0.6351**	
MLB	**0.6244**				**0.6244**	
Robotics	0.6971	**0.8174**	**0.8270**		**0.8270**	0.7874
RSC	**0.6404**				**0.6404**	
Terrorist	**0.7030**	0.6170	0.6671		0.6671	0.6232

Table 2: AUC of different algorithms on the datasets. Bold entries represents best performaning or indistinguishable from best performaning algorithm. The table shows that XGDA outperforms the competing strategy of running k-groups for long period of time across all datasets.

	N	R	T_{RP}
disjoint	500	2000	1s
EAGLE-27	9983	1926008	12m 57s
EAGLE-45	388474	162639	1m 47s
EAGLE-63	260467	103675	2m 43s
EAGLE-71	5508508	2149464	4m 43s
AUTON	120	95	1s
CiteSeer	80106	80000	2m 20s
Containers	9505	344658	36s
IOBDB	29446	3686	3m 53s
MLB	5000	2471	1m 12s
Robotics	1095	1474	2s
RSC	9500	3145	1m 19s
Terrorist	4047	5581	3s

Table 3: Running time of the fast component on the datasets. The first column is the number of entities in the dataset, the second column is the number of recrods, and the last column is the time taken to run the fast component, which is the recursive partitioning algorithm. The table shows that the recursive partitioning algorithm is very fast. It took less than five minutes for it to run on huge dataset like EAGLE-71.

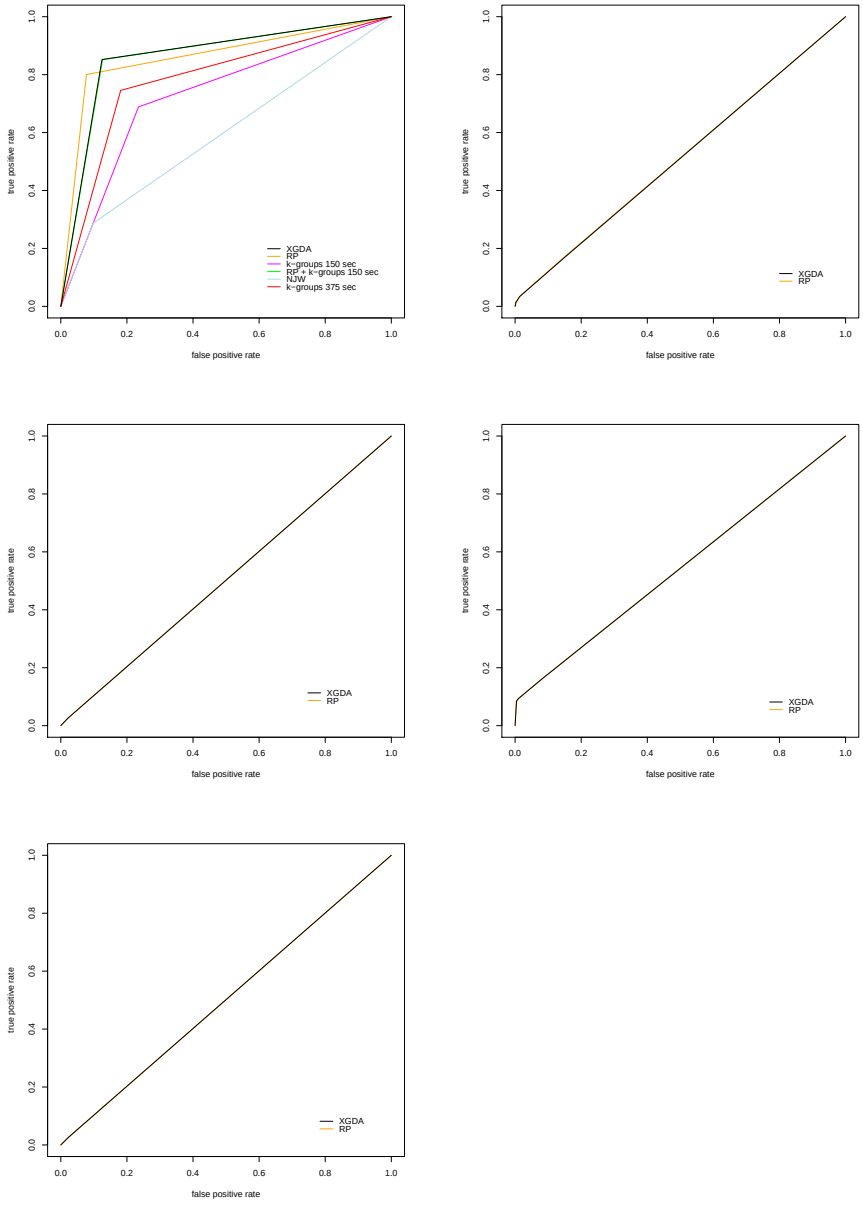

Figure 6: ROC plot of the groups detection algorithms on five synthetic datasets. Upper left: disjoint dataset. Upper right: Eagle-27 dataset. Middle left: Eagle-45 dataset. Middle right: Eagle-63 dataset. Bottom left: Eagle-71 dataset. The performance of XGDA is represented by the black curve. The ROC curve partitioning algorithm is drawn in orange. The performance of a competing group detection strategy, running k-groups for as long as the running time of XGDA, is drawn in red.

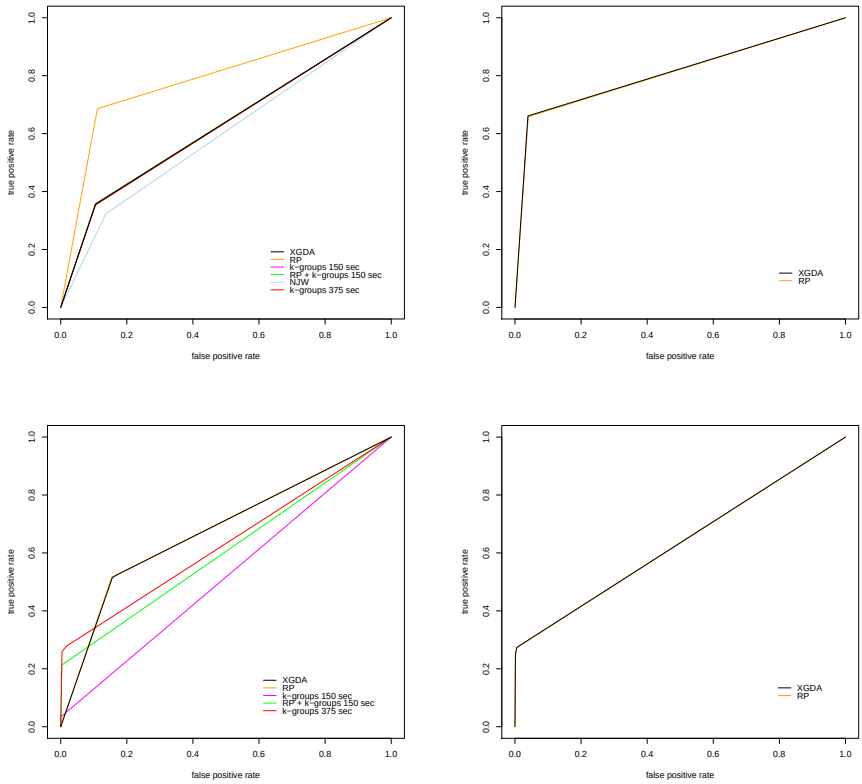

Figure 7: ROC plot of the groups detection algorithms on four real datasets. Upper left: AUTON dataset. Upper right: CiteSeer dataset. Lower left: Containers dataset. Lower right: IOBDB dataset. The performance of XGDA is represented by the black curve. The ROC curve partitioning algorithm is drawn in orange. The performance of a competing group detection strategy, running k-groups for as long as the running time of XGDA, is drawn in red.

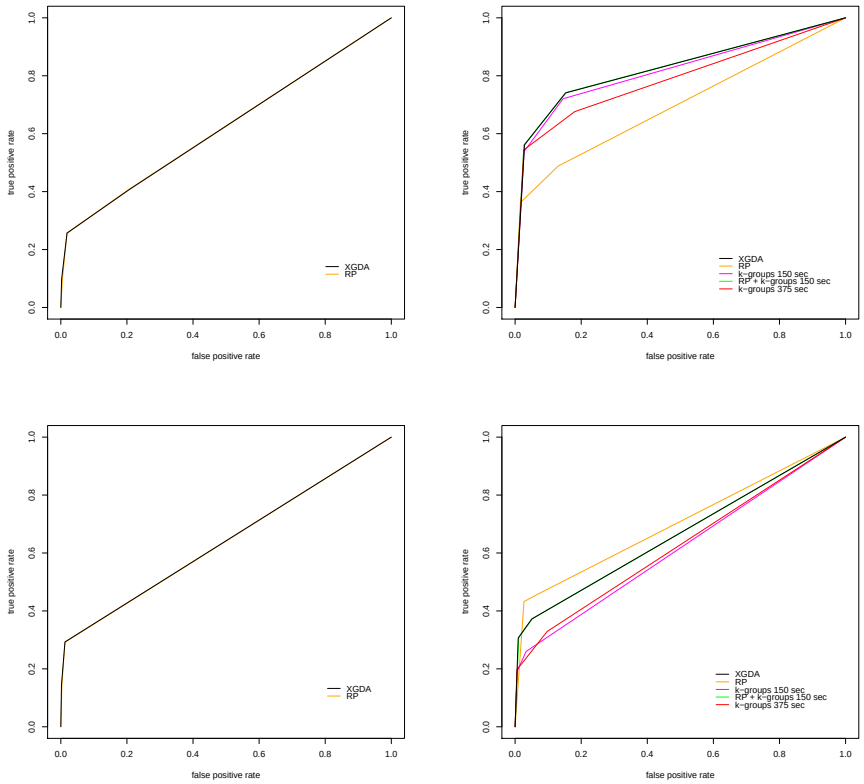

Figure 8: ROC plot of the groups detection algorithms on four real datasets. Upper left: MLB dataset. Upper right: Robotics dataset. Lower left: RSC dataset. Lower right: Terrorist dataset. The performance of XGDA is represented by the black curve. The ROC curve partitioning algorithm is drawn in orange. The performance of a competing group detection strategy, running k-groups for as long as the running time of XGDA, is drawn in red. These plot shows that XGDA outperforms the competing strategy of running k-groups for long period of time.

8 Discussion

We demonstrated that the proposed group detection algorithm XGDA outstripes the previous k-group algorithm in terms of processing time and solution accuracy. These advances may open up new application of automated group discovery system in domains that require fast and accurate large-scale transactional data analysis.

In group discovery applications, the definition of group is subjective. For example, different individuals may have different criteria in drawing the line between friends and acquaintances. Moreover, real data do not have ground truths of the underlying groups. To address the problems of subjective group definition and lack of ground truths, we proposed a group definition that associates membership in a group with a higher chance of future co-occurrence of the members. To evaluate the accuracy of predicting future co-occurrence, cross-validation method is applied. In machine learning, the definition of metric is crucial to the performance of the learning system. Therefore, exploring other group definitions may yield interesting results and may lead to alternative group discovery algorithm.

References

Chung, F. R. K. (1997). *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC.

Drummond, C. and Holte, R. C. (2000). Explicitly representing expected cost: an alternative to roc representation. In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 198–207, New York, NY, USA. ACM Press.

Fawcett, T. (2003). Roc graphs: Notes and practical considerations for data mining researchers.

He, X., Zha, H., Ding, C. H. Q., and Simon, H. D. (2002). Web document clustering using hyperlink structures. *Comput. Statist. Data Anal.*, 41(1):19–45.

Kincaid, D. and Cheney, W. (1991). *Numerical analysis: mathematics of scientific computing*, pages 257–262. Brooks/Cole Publishing Co., Pacific Grove, CA, USA.

Kubica, J., Moore, A., and Schneider, J. (2003). Tractable group detection on large link data sets. In Wu, X., Tuzhilin, A., and Shavlik, J., editors, *The Third IEEE International Conference on Data Mining*, pages 573–576. IEEE Computer Society.

Kubica, J., Moore, A., Schneider, J., and Yang, Y. (2002). Stochastic link and group detection. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 798–804. AAAI Press/MIT Press.

Ng, A. Y., Jordan, M. I., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856.

Provost, F. J. and Fawcett, T. (1997). Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Knowledge Discovery and Data Mining*, pages 43–48.

Shi, J. and Malik, J. (1997). Normalized cuts and image segmentation. In *CVPR*, pages 731–737.

A Transformation of the general eigensystem for Normalized Cut

We are solving for $\vec{x}$ the eigenvectors of the general eigensystem

$$(\mathbf{D} - \mathbf{W})\vec{x} = \lambda \mathbf{D}\vec{x} \tag{2}$$

Let $\vec{y}$ be $\mathbf{D}^{1/2}\vec{x}$.

$$\begin{aligned}
(\mathbf{D} - \mathbf{W})\vec{x} &= \lambda \mathbf{D}\vec{x} \\
(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}\vec{y} &= \lambda \mathbf{D}^{1/2}\vec{y} \\
\mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}\vec{y} &= \lambda \vec{y} \\
(I - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\vec{y} &= \lambda \vec{y} \\
\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\vec{y} &= (1 - \lambda)\vec{y} \qquad (3)
\end{aligned}$$

Therefore the second smallest eigenpair of the original eigensystem (2) is related to the second largest eigenpair of the eigensystem (3) through a matrix multiplication by $\mathbf{D}^{-1/2}$.