

K-groups: Tractable Group Detection on Large Link Data Sets

Jeremy Kubica

Andrew Moore

Jeff Schneider

CMU-RI-TR-03-32

August 2003

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

© Carnegie Mellon University

Abstract

Discovering underlying structure from co-occurrence data is an important task in many fields, including: insurance, intelligence, criminal investigation, epidemiology, human resources, and marketing. For example a store may wish to identify underlying sets of items purchased together or a human resources department may wish to identify groups of employees that collaborate with each other.

Previously Kubica *et. al.* presented the group detection algorithm (GDA) - an algorithm for finding underlying groupings of entities from co-occurrence data. This algorithm is based on a probabilistic generative model and produces coherent groups that are consistent with prior knowledge. Unfortunately, the optimization used in GDA is slow, making it potentially infeasible for many real world data sets. For example, in the co-publication domain the MEDLINE database of medical publications alone contains over 2 million papers published within just a 5 year period, 1995-1999 [14].

To this end, we present k-groups - an algorithm that uses an approach similar to that of k-means (hard clustering and localized updates) to significantly accelerate the discovery of the underlying groups while retaining GDA's probabilistic model. In addition, we show that k-groups is guaranteed to converge to a local minimum. We also compare the performance of GDA and k-groups on several real world and artificial data sets, showing that k-groups' sacrifice in solution quality is significantly offset by its increase in speed. This trade-off makes group detection tractable on significantly larger data sets.

Contents

1	Introduction	1
2	Notation	2
3	The Group Detection Algorithm (GDA)	2
3.1	Generative Model and Probabilities	3
3.2	Optimization	5
4	Additional Notation	5
5	The K-groups Algorithm	5
5.1	Determining Group Ownerships	6
5.2	Updating the Groups	6
5.3	Demographics Information	7
5.4	Proof of Covergence	7
5.5	Avoiding Local Minima	8
6	Comparison with GDA on Real World Data	9
6.1	Data Sets	9
6.2	Results	10
7	Comparison with GDA on Artificial Data	11
7.1	Data and Evaluation	12
7.2	Results	12
8	Related Work	14
9	Conclusions	15

1 Introduction

Co-occurrence data is an increasingly important and abundant source of data. This type of data has long been important in the social sciences, marketing, and the government intelligence community. More recently it has become a topic of increasing interest in computer science for such tasks as: analysis of text documents [2, 4, 10], analysis of internet content and structure [8], and collaborative filtering [1, 3, 16].

In one general form the input consists of a series of *links*. Each link is a set of entities that have been joined by some event or relation. Table 1 shows example links from the co-publication domain. Each paper defines a single link containing its authors. Similarly, links could be formed from such events/relations as: co-occurrences in purchases (market basket analysis), co-occurrences at meetings, or direct phone conversations. It is important to appreciate that our definition of a link may vary from other usages, such as a link on a webpage. We do not assume any directionality or restrict links to contain a fixed number of entities. Depending on the domain links may contain noise or even be pure noise (sets of unrelated entities).

One important task is the identification of underlying structure buried within large amounts of noisy link data. Many algorithms have been proposed to find different types of structure from this type of data. We discuss these algorithms and their relation to this work in the related work section. Below we examine the group detection algorithm (GDA), an algorithm that attempts to find underlying groups of entities given link and demographic data [13]. GDA is based on a probabilistic generative model, in the form of a Bayesian network, that assumes that links are the result of underlying groups. Qualitatively GDA has been found to produce groups that are consistent with prior knowledge. Unfortunately GDA makes use of heuristic optimization techniques that may be slow, making it potentially infeasible for many real world data sets that may be large. For example, in the co-publication domain the MEDLINE database of medical publications alone contains over 2 million papers published within just a 5 year period, 1995-1999 [14]. Domains such as criminal intelligence and market basket analysis present the possibility of even more daunting amounts of data.

To this end we propose k-groups, an algorithm that uses a probabilistic model similar to GDA but uses localized updates to improve both speed and convergence properties. We show that k-groups is guaranteed to converge to a local minimum and compare its performance to that of GDA on several real world and artificial data sets. We show that k-groups' sacrifice in solution quality is significantly offset by its increase in speed. This trade-off makes group detection tractable on significantly larger data sets.

The rest of this paper is organized as follows. In section 2 we introduce our notation. In section 3 we formally describe the group detection algorithm and input data sets. In section 4 we provide additional notation. In section 5 we formally describe the k-groups algorithm and prove that it converges to a local minimum in a finite number of steps. Sections 6 and 7 examine the relative performance of GDA and k-groups on real world and artificial data respectively. Finally, related work and its relation to GDA and k-groups is discussed in section 8.

PAPER ID	SET OF ENTITIES
KGROUPS	{J KUBICA, A MOORE, J SCHNEIDER}
GDA	{J KUBICA, A MOORE, J SCHNEIDER, Y YANG}
CGRAPH	...

Table 1: Example links from a co-authorship domain. Each row (paper) is a single link containing its authors as entities.

ENTITY	TITLE	AFFILIATION	...
J KUBICA	STUDENT	CMU	...
A MOORE	PROFESSOR	CMU	...
...

Table 2: Example demographics from a co-authorship domain.

2 Notation

We begin our discussion by introducing some notation. The input data is assumed to contain N_P unique entities. We denote the set of all entities as ξ and a single entity as $e_i \in \xi$. The input data consists of N_L links, each of which is a subset of entities. A single link is denoted $L_i \subseteq \xi$ and contains $|L_i|$ entities. The entire set of links is denoted as LD . Finally we are attempting to find K groups. We denote a group as $g_i \subseteq \xi$ and the set of all current groups as G . The subscripts on e , L , and g are often left off for clarity.

As stated above, links can contain noise or be entirely noise. We say that a link contains noise if the link L was generated by some group g and there is some entity $e \in L$ such that $e \notin g$. In this case we break down the link size as $|L| = M_R + M_G$, where M_G is the number of entities in the link that are also in the generating group g and M_R is the number of entities that are random noise (i.e. not in g).

Below we also make extensive use of the fact that the logarithm is monotonic. Therefore maximizing $\log(X)$ is equivalent to maximizing X itself.

3 The Group Detection Algorithm (GDA)

The group detection algorithm (GDA) uses maximum likelihood estimation to find groupings of entities given two input data sets [13]. The first data set is the demographics data set, which contains all the entities under consideration and their demographic information. The word “demographic” should not be interpreted too narrowly as it can include any information available about that entity. Table 2 shows example demographics from the co-publication domain, including the author’s title and affiliation. The second data set is the link data set, which is just a set of records specifying observed links. This data set can also be viewed as a sparse $N_L \times N_P$ matrix where there is one column for each entity and one row for each link. The entries in this matrix indicate whether a given entity is in a given link.

The below formulation of GDA follows the one given by Kubica *et. al.* and uses the same models [13].

3.1 Generative Model and Probabilities

The basis of GDA is a probabilistic generative model shown in Figure 1. The model is a Bayesian network where the ovals indicate parameters to be learned from data. The five primary components of this model are:

1. the *demographics data set (DD)* described above;
2. the *link data set (LD)* described above;
3. the *chart (CH)*, which indicates which entities belong to which groups. This data structure can be pictured as a sparse $N_P \times K$ matrix where the columns indicate the groups and the rows indicate the entities. Note that unlike traditional clustering models, an entity can simultaneously be a full member of several groups (have a “one” in several columns);
4. the *demographic model (DM)*, which defines a recipe for placing a entity in a group based on demographic information; and
5. the *link model (LM)*, which defines a recipe for link generation. The link model contains two parameters for each link type: the probability a link of that type is completely random (innocent), P_I , and the probability that an entity in a link of that type is noise (random), P_R . These probabilities can be learned or specified by the user. Below we treat these probabilities as given.

The above components provide a recipe for generating the data from the model. Using the decomposition provided by the Bayesian network in Figure 1 we can examine the steps of data generation:

$$P(DM, DD, CH, LM, LD) = P(CH|DM, DD)P(LM)P(DM)P(DD)P(LD|LM, CH) \quad (1)$$

The first step of data generation is the formation of groups. Each group has its own demographics model, a probability density function that indicates the probability an entity is a member of the group given its demographic information. Each entity/group membership is considered with the demographics model indicating the probability the entity is a member of the group given its demographics. $P(CH|DM, DD)$ represents the probability that all of the group memberships in the chart occur, or do not occur, given all the demographic information and the demographic model for each group.

The second step of data generation is the formation of links from the link model and groups. The model assumes that links are generated individually by choosing a group g and uniformly sampling members from it. The amount of noise in a link is determined by the link model. Again, we say that a link L contains noise if it was generated g and there is some entity $e \in L$ such that $e \notin g$. During link generation, this corresponds to choosing each entity in the link directly from g or with some probability P_R as noise (from $\xi - g$). Consequently links are noisy reflections of the underlying groups. In

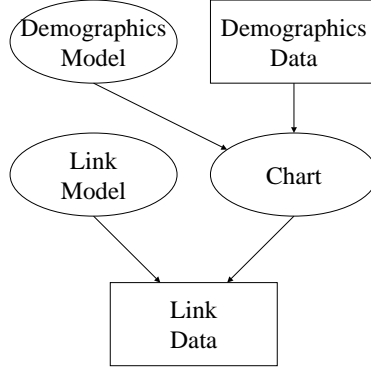


Figure 1: Probabilistic model of group membership and link generation.

addition, with probability P_I a link can be completely random. A completely random link is one where all of its entities are chosen uniformly from ξ . Since the links are i.i.d., we can calculate $\log P(LD|LM, CH)$ as:

$$\log P(LD|LM, CH) = \sum_{i=1}^{N_L} \log P(L_i|LM, CH) \quad (2)$$

where $P(L_i|LM, CH)$ is the probability of an individual link given the link model and the chart. If we make the simplifying assumption that the priors for each group generating a link are equal, we can define $P(L|LM, CH)$ as:

$$P(L|LM, CH) = \left(\frac{P_I}{\binom{N_P}{|L|}} + \frac{1 - P_I}{K} \sum_{g \in G} P(L|g, LM) \right) \quad (3)$$

$P(L|g, LM)$ is the probability of link L given that group g generated it. Using the assumption of uniform sampling (without replacement), $P(L|g, LM)$ is defined as:

$$P(L|g, LM) = \left(\frac{(P_R)^{M_R} (1 - P_R)^{M_G} \binom{M_G + M_R}{M_R}}{\binom{|g|}{M_G} \binom{N_P - |g|}{M_R}} \right) \quad (4)$$

Again M_G is the number of people in the link that are members of group g and M_R is the number of people in the link that are not members of g .

One important approximation is shown in (5). This creates the concept of a group “owning” a link and both allows the distribution of the logarithm and can lead to significant computational speedups.

$$\log(P(L|LM, CH)) \approx \log \left(\max \left(\frac{P_I}{\binom{N_P}{|L|}}, \frac{1 - P_I}{K} \max_g (P(L|g, LM)) \right) \right) \quad (5)$$

Finally, we do not consider the contributions of $P(LM)$, $P(DD)$ or $P(DM)$ to (1). Formally, we consider all DM s and LM s to be equally likely and accordingly ignore their contributions to (1). $P(DD)$ can be ignored since it will be the same for all parameters.

3.2 Optimization

The actual maximum likelihood estimation is accomplished by using noisy hill climbing, a heuristic optimization method. The goal is to find the chart, demographics model, and possibly link model that maximize the likelihood as given by the entire Bayesian network in Figure 1. This is done by trying different moves (adding an entity to a group or removing an entity from a group) and evaluating the change in the loglikelihood. The max approximation shown in (5) was used to achieve significant computational speedups by restricting the number of groups for which the likelihood had to be evaluated when evaluating a move.

4 Additional Notation

Before we introduce the k-groups algorithm, we first introduce some additional notation and make a few additional observations for use below. We define the *world group*, $G_W = \{e_j : 1 \leq j \leq N_P\}$, to be the group of all entities. Thus the probability of a completely random link becomes:

$$P(L, L \text{ is Random} | LM) = \frac{P_I}{\binom{N_P}{|L|}} = P_I P(L | G_W, LM) \quad (6)$$

We can then define $\Omega = \{g_1, \dots, g_K, G_W\}$ as the set of all possible link generators/owners. In other words, a link can be created by any of the K groups or be completely random. Finally, we observe that the probability of a group, $g \in \Omega$, having generated a link can be written as:

$$P(L, g | LM) = \begin{cases} \frac{P_I}{\binom{N_P}{|L|}} & \text{if } g = G_W \\ \frac{(1-P_I)P(L|LM,g)}{K} & \text{if } g \neq G_W \end{cases} \quad (7)$$

5 The K-groups Algorithm

The k-groups algorithm focuses on the task of learning the underlying groups directly from the link data. That is to say that it optimizes $P(LD | LM, CH)$ and does not consider the demographics information in the inner loop. Intuitively k-groups optimizes $P(LD | LM, CH)$ in a fashion similar to that of the k-means algorithm. Specifically, it alternates between two greedy steps:

1. Determine group ownerships given the groups.
2. Determine the groups given group ownerships.

The key difference between k-groups and k-means algorithm is that groupings in the k-groups algorithm are not disjoint and can in fact have significant overlap. This approach can still be adapted by noting that the max approximation in (5) forces groups to “own” links. If we define LG_g to be the set of links “owned” by a group $g \in \Omega$, we can re-express $\log(P(LD|LM, CH))$ as:

$$\begin{aligned} \log(P(LD|LM, CH)) &= \sum_{L \in LD} \max_{g \in \Omega} \log(P(L, g|LM)) \\ &= \sum_{g \in \Omega} \log(P(LG_g \wedge g|LM)) \end{aligned} \quad (8)$$

where

$$\log(P(LG_g \wedge g|LM)) = \sum_{L \in LG_g} \log(P(L, g|LM)) \quad (9)$$

In step 1 we can partition the *links* into sets owned by the groups so as to optimize $\log(P(LD|LM, CH))$ and in step 2 we can assign entities to groups so as to optimize $\log(P(LD|LM, CH))$. The k-groups algorithm becomes:

Until convergence:

1. For each link, determine which group owns it.
2. For each group g , determine which entities form g so as to optimize $P(LG_g|g, LM)$.

5.1 Determining Group Ownerships

The first step of the k-groups algorithm is to determine which links each group owns. This can be done quickly with a single linear scan through the links using the approximation in (5):

$$LG_g = \{L : g = \underset{g' \in \Omega}{\operatorname{argmax}} P(L, g'|LM)\} \quad (10)$$

5.2 Updating the Groups

The second step of the k-groups algorithm is to find the “optimal” groups given the links they own. Since we are considering only the links owned by the current group we wish to optimize $P(LG_g \wedge g|LM)$. Note that by combining equations (4) and (9) we can examine the effect of adding or removing a single entity to the group on $\log P(LG_g, g|LM)$. Let $\Delta_g^A(e)$ be the change in $\log P(LG_g, g|LM)$ if we add entity e to group g and $\Delta_g^R(e)$ be the change in $\log P(LG_g, g|LM)$ if we remove entity e from group g . Then by algebra:

$$\Delta_g^A(e) = \begin{cases} \sum_{L \in LG_g} \left[\log \left(\frac{|g|-M_G+1}{N-|g|-M_R} \right) - \log \left(\frac{|g|+1}{N-|g|} \right) \right] & \text{if } e \notin g \\ + \sum_{L \in LG_g: e \in L} \left[\log \left(\frac{1-P_R}{P_R} \right) - \log \left(\frac{|g|-M_G+1}{N-|g|-M_R} \right) \right] & \\ 0 & \text{if } e \in g \end{cases} \quad (11)$$

$$\Delta_g^R(e) = \begin{cases} \sum_{L \in LG_g: e \in L} \log\left(\frac{P_R}{1-P_R}\right) - \sum_{L \in LG_g} \log\left(\frac{N-|g|+1}{|g|}\right) & \text{if } e \in g \\ + \sum_{L \in LG_g: e \notin L} \log\left(\frac{N-|g|-M_R+1}{|g|-M_G}\right) & \\ 0 & \text{if } e \notin g \end{cases} \quad (12)$$

Using these observations we can define a second, inner greedy procedure to optimize $P(LG_g|g, LM)$:

1. For each $e \in LG_g$ calculate $\Delta_g^A(e)$ and $\Delta_g^R(e)$.
2. If there exists some entity e such that $\Delta_g^A(e) > 0$ (or $\Delta_g^R(e) > 0$) then add (or remove) the entity that would lead to the greatest improvement.
3. If $\Delta_g^A(e) \leq 0$ and $\Delta_g^R(e) \leq 0 \forall e$, terminate.

This procedure greedily adds/removes entities until it no longer results in an improvement. This approach has three major computational advantages. First, we do not have to recalculate $P(LG_g, g|LM)$ for each change that we wish to evaluate. This allows us to rapidly try each entity exhaustively. Second, k-groups only needs to consider a subset of links for each group. Finally, k-groups only needs to consider a subset of entities for each group. Specifically, k-groups only needs to consider adding entities that are not in the group and are in *at least one* link in LG_g and removing entities that are currently in the group.

Finally it is important to appreciate that this search is localized and therefore may not find groups that would optimize the overall likelihood of the data. It is relatively easy to construct a case where $\Delta_g^A(e) < 0$, but adding e to g will increase the overall likelihood by causing g to own new links. Such a move will not be found by the above procedure.

5.3 Demographics Information

The k-groups algorithm does not currently make use of demographic information during the course of the localized updates. Rather it can learn a demographic model *after* it has finished building the groups. It is possible that incorporating the demographic information into the localized updates may lead to further improvements in performance. We leave this investigation as future research.

5.4 Proof of Covergence

One important advantage of the k-groups algorithm is that we can show that it converges to a local optimum in a finite number of steps. Formally we state this as:

Theorem: Upon termination $P(LD|LM, CH)$ can not be improved by any one of the following three changes: adding a single entity to a group, removing a single entity from a group, or assigning a link to be owned by another group.

Intuitively this holds because steps 1 and 2 above use these three changes to maximize $P(LD|LM, CH)$. We provide a formal proof below. We use as the termination criteria that *neither* step results in an improvement of $P(LD|LM, CH)$.

Proof: First we prove that upon termination $P(LD|LM, CH)$ can not be improved by any one of the above three changes. After step 1, the links have been assigned to groups so as to optimize $P(LD|LM, CH)$. This follows directly from the update given in (10). Thus after step 1 if the groups remain fixed, no change in link ownership will improve $P(LD|LM, CH)$. Similarly step 2 maximizes $P(LD|LM, CH)$ by adding entities to or removing entities from groups. This follows directly from the update rule presented in section 5.2. Thus after step 2 if the link assignments remained fixed, there is no single entity that can be added to or removed from a group so as to improve $P(LD|LM, CH)$. Further, both steps will only make a change if it leads to an *improvement* in $P(LD|LM, CH)$. Thus if neither step results in an improvement of $P(LD|LM, CH)$, then neither step resulted in a change (both the groups and link assignments remain fixed) and the algorithm is at a local optimum.

Next we prove that this optimum will be reached after a finite number of steps. Consider each possible set of link ownerships and group memberships as a state. There are a finite number of states, $O(2^{(N_p K)} K^{N_L})$. As shown above, each iteration of k-groups (steps 1 and 2) increases $P(LD|LM, CH)$ or leads to termination. Thus each iteration of k-groups transitions to a better state or terminates. Since there are a finite number of states, the algorithm will terminate in a finite number of steps. Q. E. D.

While k-groups is guaranteed to converge in a finite number of steps, this number could be as large as $O(2^{(N_p K)} K^{N_L})$. Despite this upper bound, we have empirically found that the algorithm often converges relatively quickly.

5.5 Avoiding Local Minima

Although k-groups is guaranteed to converge to a local minimum, early results showed that, as often is the case, this minimum is not likely to be the global minimum. To combat this, we use two simultaneous strategies for “getting unstuck” from local minima. Both serve to perturb the solution before the inner k-groups loop is rerun. Consequently, each convergence becomes a single iteration of the larger k-groups algorithm.

The first strategy is similar to the approach used in Split-Merge EM [18]. Specifically two groups are chosen such that merging them causes the smallest decrease in likelihood. We merge two groups by merging their sets of links and creating a new single group to account for these links. The two groups are merged into a single group,

DATA SET	N_P	N_L	K
LAB	115	94	20
INSTITUTE	456	1738	100
DRINKS	136	5325	50
MANUAL	4088	5581	25
CITSEER	104801	181395	50

Table 3: Summaries of the data sets.

which replaces one of the original groups. The other group is simply replaced by a group filled with random entities.

The second strategy is to add a small amount of noise to a random number of groups in the solution. Specifically with some fixed probability, P_g , we chose to add noise to a group by iterating through the N_P entities and adding/ removing them with some fixed probability, P_{flip} . In the below experiments we chose these probabilities heuristically as $P_g = \frac{2}{K}$ and $P_{flip} = \frac{2.5}{N_P}$.

6 Comparison with GDA on Real World Data

The key advantage of the k-groups algorithm is that by using localized updates it finds “better” solutions faster than the simple heuristic optimization used by GDA. To demonstrate this performance we tracked the training set log-likelihood versus the time spent in the process for both algorithms.

6.1 Data Sets

To test the k-groups algorithm, we used a variety of real world data sets. These data sets are described below and are summarized in Table 3. These data sets include filtered versions of the data sets used in [12] and will be available at <http://www.autonlab.org/>

1. The *Lab* data consists of co-publication links for members of our research lab and includes as entities all authors.
2. The *Institute* data is a set of links of three different types (co-publication, common research interest, and advisor/advisee) that was collected from publicly available data listed on Carnegie Mellon University Robotic Institute’s web-pages.
3. The *Citeseer* data is a collection of co-publication links from the Citeseer online library and index of computer science publications. Each publication served as a single link containing its authors. Since entities were represented by first initial and last name, a single name could correspond to multiple authors.
4. The *Drinks* data set consists of a series of popular bar tending recipes found on various websites. Each link consisted of a list of all ingredients (entities) contained in that drink.

DATA SET	K-GROUPS 1 ITR. LL	K-GROUPS TIME	GDA TIME	TIMES SPEEDUP
LAB	-726	0.13	145	1163.2
INSTITUTE	-18847	16.00	667	41.7
DRINKS	-42664	23.25	349	15.0
MANUAL	-86509	9.38	5033	536.9
CITSEER	-4763400	1319.00	N/A	N/A

Table 4: The average loglikelihood after one iteration of k-groups and the average time (in seconds) for k-groups and GDA to reach that loglikelihood.

5. The *Manual (Webpages)* data is a set of links created by a human who manually read a set of public web pages and news stories related to terrorism and subjectively linked entities mentioned in the articles. Each link was created by hand from a single relation, such as `memberOf` or `funded`, and contained the entities for which this relation was mentioned.

6.2 Results

Both algorithms were run on all data sets using identical parameter settings, $P_I = 0.2$ and $P_R = 0.2$. The parameters were set heuristically so as to allow a reasonable amount of noise. Group sizes varied among the data sets as: 20 for the Lab data, 100 for the Institute data, 50 for the groups data, 25 for the manual data, and 50 for the citeseer data.

Figure 2 shows the results of the runs. Since both GDA and k-groups are randomized algorithms, they were run multiple times on each data set. For the Lab, Institute and Drinks data each algorithm was run 24 times. For the Manual and Citeseer data they were each run 8 times and 1 time respectively. To this end, at each time each plot indicates the mean loglikelihood and the bounds of the 95% confidence interval on performance. No confidence intervals were included on the Citeseer results.

As the results illustrate, k-groups can offer a significant speedup. As the size of the data sets increase (number of links and entities) this speedup can become more pronounced and also more important. For example, on the Citeseer data k-groups was able to converge to a local minima in under 22 minutes that was better than any solution found by GDA within 24 hours.

The above results lead to another natural question: “How good is the first local minima found by k-groups?” For each data set we examined the average loglikelihood after a single iteration of k-groups (after it converges to the first local minima) and the average time it took to execute this iteration. This was then compared to the GDA results. Specifically we asked: “How long on average does it GDA to reach this loglikelihood?” The results are shown in Table 4. Entries marked with *N/A* indicate that GDA did not find a solution as good as the one from kGroup’s first iteration in the time allotted. As shown a single iteration often performs relatively well and is *much* faster than finding a equally good solution using the heuristic optimization technique.

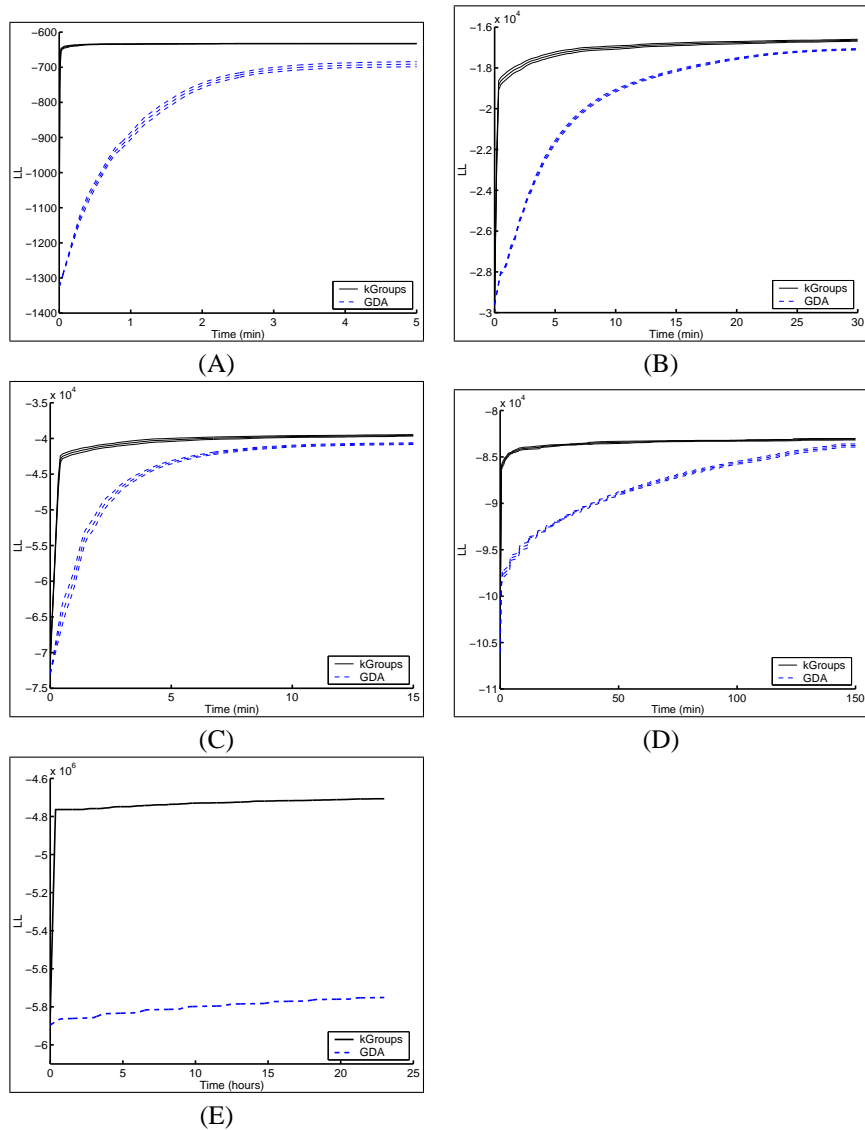


Figure 2: Loglikelihood versus time for (A) Lab, (B) Institute, (C) Drink, (D) Manual, and (E) Citeseer data. Larger values on the y-axis indicate better performance at a given time.

7 Comparison with GDA on Artificial Data

The above results demonstrate a significant computational advantage for the k-groups update. To further explore this, we used artificial data to examine how the algorithms perform as various factors (number of links or groups) change.

7.1 Data and Evaluation

The data consisted of links generated from the above GDA model and known groups. At the start of each test the groups were generated at random such that each group had an average of 10 entities. Links were then created using a fixed link model with $P_R = 0.2$ and $P_I = 0.2$. The number of links, groups, and entities were given for each run.

Since we knew the true groups that generated the data, we can evaluate the learned groups by asking: “If we had to represent the true groups as the learned groups, how many errors would we make?” In other words, for each true group g we found the learned group \hat{g} that best approximated it and counted the number of errors. The sum of these errors was the overall score of the learned groups. Formally, we define the error between two groups as the Hamming distance:

$$ERR(g, \hat{g}) = |g| + |\hat{g}| - 2|g \cap \hat{g}| \quad (13)$$

and the overall error as:

$$ERR = \sum_{k=1}^K \min_{\hat{g}} ERR(g_k, \hat{g}) \quad (14)$$

7.2 Results

It should first be noted that on most data sets it is likely that GDA will eventually outperform k-groups. K-groups uses a localized update and may never find the global optimum. In contrast, GDA uses an update that tests a variety of moves and therefore will most likely eventually find the global optimum. Despite this, the k-groups algorithm may often be able to find good solutions significantly faster than GDA and therefore produce good approximations in a tractable manner. Accordingly a key factor examined below is the time it takes GDA to “catch-up” to k-groups. We refer to this time as the *catch-up time* and define it formally as the latest time when k-groups had an average performance better than that of GDA. A larger catch-up time indicates better initial performance of k-groups relative to GDA.

Increasing the number of groups should negatively effect both algorithms simply because it adds additional information to be learned. Figures 3 (A), (B), and (C) demonstrate the effect of the number of groups on the algorithms, illustrating the average error versus the wall clock time with 20, 50, and 100 groups respectively. All of the runs use 500 entities and 10000 links. As expected, increasing the number of groups negatively impacts both algorithms. Further, increasing the number of groups *increased* the catch-up time. This indicates that as the number of groups increases so does k-groups’ initial advantage.

In contrast to the number of groups, both GDA and k-groups can be expected to benefit from the existence of additional links, because the links provide more data. Figures 3 (D), (E), and (F) demonstrate the effect of the number of links on the algorithms, illustrating the average error versus the wall clock time with 5000, 20000, and 40000 links respectively. All of the runs use 500 entities and 50 groups. As expected, the number of links positively affected both algorithms performances. Again, increasing the number of links *increased* the catch-up time. This observation agrees with the

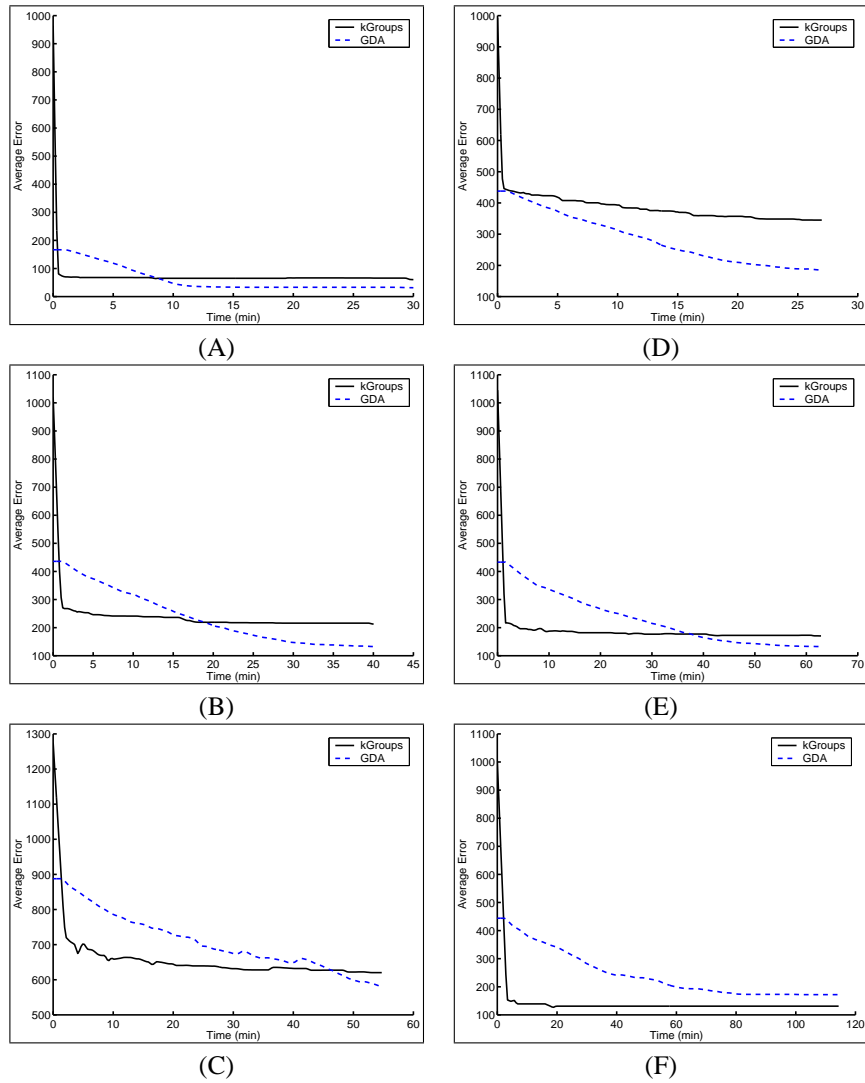


Figure 3: The average error versus time as the number of groups and number of links is varied. (A)-(C) show results for a varying number of groups: (A) 20, (B) 50, and (C) 100. (D)-(F) show results for a varying number of input links: (D) 5000, (E) 20000, and (F) 40000. Results indicate errors; lower values are better.

key performance gain of k-groups. Specifically, k-groups uses hard clustering and localized updates. Therefore we would expect that the algorithm would scale better as the number of links increases.

From the above results we can see that while increasing either of two key factors (number of groups and number of links) can have different effects on the performance

of both algorithms as a whole, they both lead to better performance of k-groups relative to that of GDA. We would expect that as these factors continue to grow, the k-groups algorithm will scale more gracefully than GDA.

8 Related Work

The k-groups algorithm is an improvement of the GDA algorithm [13]. In addition there are a variety of similar algorithms that extract different types of structure from link data, including: underlying structural graphs [11, 12, 14], hubs and authorities [8], latent class labels [4, 10], latent dirichlet models [2], and other Bayesian models [3, 5, 7, 17]. We discuss these models and their relations to GDA in more detail below.

Underlying structural graph algorithms attempt to build a graph model of the data [11, 12, 14]. The nodes represent entities and the edges capture pairwise relations between the entities. Unlike GDA, all underlying structure is pairwise and there is no attempt to discover underlying groups.

Hubs and authorities algorithms find structure and “web communities” from directional link information, such as the internet links [8]. Where as these algorithms make explicit use of the directionality, we examine data without directionality, such as co-publication information.

Latent class label algorithms and latent dirichlet models (LDA) are similar to GDA [2, 4, 10]. They use an underlying generative model to explain link creation and the latent class variable can be viewed as a group. In fact Cohn and Hoffman present a model for document connectivity, where document terms play a role similar to our demographic information [4]. Despite this, the type of “group” returned by these algorithms is different. They are effectively clustering the links themselves and using probabilistic models of the entities’ occurrences in the links to measure the likelihood that a group (or groups) generated the link. In contrast, by design GDA returns groupings of entities with definite memberships. Although it may be possible to adapt these methods to return groupings of entities from their probabilistic models, these algorithms are not trying to find this type of grouping and the best approach to this adaptation is not immediately clear.

Finally, other probabilistic approaches, such as probabilistic relational models and Bayesian networks, have been used to model this type of data [3, 5, 7, 17]. For example, Tasker *et. al.* propose a clustering approach based on the relational aspects of the data [17]. While they also focus on clustering the entities, they use a latent class model. This model restricts assignments such that an entity can only be a “full” member of a single group. Therefore unlike GDA this approach does not allow an entity to fully belong to several different groups.

Although GDA’s approach is similar to some of the algorithms mentioned above, GDA is inherently trying to extract a different type of structure from the data. Specifically, GDA tries to extract underlying groupings of entities with definite group membership. Our model is thus designed to easily and directly capture the group membership nature of the data, including the fact that a person can be a member of many groups.

The k-groups algorithm is similar in approach to, and inspired by, the k-means

algorithm [6]. Both algorithms use a “hard clustering” approach for assigning group membership and use only these member points to update the group. Despite this similarity, the two algorithms differ significantly on their domains and optimization criteria. In addition, the use of a split/merge operation was proposed by Ueda *et. al.* for use in EM optimization of mixture models [18].

When rephrased as the problem of assigning link ownerships to groups, our work becomes similar to approaches that attempt to cluster this type of information by clustering the links themselves [2, 9, 15]. There are several key differences between these approaches and our own. First, we are using a relatively novel underlying generative model. Second, we are primarily interested in the resulting “clusters” of entities, which we restrict to be hard clusters representing group memberships. Finally, k-groups uses a novel greedy approach that is designed for both this generative model and the “hard”, but nonexclusive, assignments of entities to groups.

9 Conclusions

Above we presented k-groups, an algorithm that uses localized updates to improve both speed and convergence properties while still using GDA’s probabilistic model. We motivated the derivation of the algorithm using the same properties as k-means and showed k-groups is guaranteed to converge to a local minimum. Finally we compared k-groups performance to that of GDA on both real world and artificial data sets, showing that k-groups’ sacrifice in solution quality is significantly offset by its increase in speed.

There are several remaining questions that we plan to investigate in future research. The first is to learn K while learning the underlying groups by using a measure such as AIC or BIC. The second is to incorporate the use of demographics information into the localized update steps.

Finally, the results on the artificial data sets indicate that given enough time GDA will eventually catch-up to and surpass k-groups. This suggests several hybrid approaches. K-groups can be used to do initial optimization and the learned chart can then be feed into GDA for further refinement. Additionally, it might be beneficial to alternate the k-groups and GDA optimization step. Thus, in future research we hope to investigate how best to combine these optimization schemes.

Acknowledgements

Jeremy Kubica is supported by a grant from the Fannie and John Hertz Foundation. This research is supported by DARPA under award number F30602-01-2-0569. The authors would like to thank Alex Gray for his helpful comments and suggestions. The authors would also like to thank Steve Lawrence for making the Citeseer data available.

References

- [1] D. Billsus and M. Pazzani. Learning collaborative information filters. In *Proc. 15th International Conference on Machine Learning*, pages 46–54, 1998.
- [2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.
- [3] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *The Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.
- [4] D. Cohn and T. Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In T. Leen, T. Dietterich, and V. Tresp, editors, *NIPS13*, pages 430–436. MIT Press, 2001.
- [5] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309. Morgan Kaufmann Publishers, 1999.
- [6] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Communications and Information Theory. Kluwer Academic Publishers, Norwell, MA, USA, 1992.
- [7] L. Getoor, E. Segal, B. Taskar, and D. Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI01 Workshop on Text Learning: Beyond Supervision*, August 2001.
- [8] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proc. 9th ACM Conference on Hypertext and Hypermedia*. ACM, 1998.
- [9] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.
- [10] T. Hofmann. Probabilistic latent semantic analysis. In *Proc. of Uncertainty in Artificial Intelligence*, Stockholm, 1999.
- [11] H. A. Kautz, B. Selman, and M. A. Shah. The hidden web. *AI Magazine*, 18(2):27–36, 1997.
- [12] J. Kubica, A. Moore, D. Cohn, and J. Schneider. Finding underlying connections: A fast graph-based method for link analysis and collaboration queries. In *ICML2003*, pages 392–399. AAAI Press, 2003.
- [13] J. Kubica, A. Moore, J. Schneider, and Y. Yang. Stochastic link and group detection. In *AAAI*, pages 798–804. ACM Press, Jul 2002.
- [14] M. E. J. Newman. Who is the best connected scientist? a study of scientific coauthorship networks. *Phys. Rev.*, 64(016131; 016132), 2001.
- [15] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
- [16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proc. of ACM Conference on Computer Supported Cooperative Work*, pages 175–186, 1994.
- [17] B. Taskar, E. Segal, and D. Koller. Probabilistic clustering in relational data. In *Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–876, Seattle, Washington, Aug. 2001.
- [18] N. Ueda, R. Nakano, Z. Ghahramani, and G. E. Hinton. SMEM algorithm for mixture models. *Neural Computation*, 12(9):2109–2128, 2000.