Clustering from Constraint Graphs

Ari Freund* Dan Pelleg* Yossi Richter*

Abstract

In constrained clustering it is common to model the pairwise constraints as edges on the graph of observations. Using results from graph theory, we analyze such constraint graphs in two contexts, both of immediate value to practitioners. First, we explore the issue of constraint noise under several intuitive noise models. We apply results from random graph theory, which facilitate the analysis of finite-sized graphs and realistic data partitions and noise levels, to obtain a quantification of the effect noisy edges may have on *any* constrained clustering algorithm under a set of commonly-used assumptions. We also demonstrate the dangers in the common practice of connected-component constraint set augmentation, when used in the presence of noise. Second, we describe two practical randomized algorithms that estimate the number of induced clusters using only a small number of constraints. We conclude with an experimental evaluation that shows the effect of noise on common UCI data sets, as well as some aspects of the behavior of our algorithms.

1 Introduction

For a long time, clustering was associated solely with unsupervised learning. The absence of supervision (in the form of explicit class labels for data instances) meant wider applicability of clustering algorithms. But it also made for a weakness, because it was impossible to offset the clustering algorithm's internal bias and direct it towards a particular type of solution. Recently, the discipline of constrained clustering has emerged as a way to provide this kind of direction without requiring explicit labeling of data points.

In constrained clustering the general form of the clustering problem is augmented with a set of constraints that the clustering algorithm should consider. The most popular form of constraints are instance-level constraints, in particular, *must-link* (ML) constraints to indicate that a pair of input points need to be in the same output cluster, and *cannot-link* (CL) constraints to indicate the opposite. This is the formulation of greatest interest, and the only one we explore in this paper. Recall that traditional clustering criteria keep intra-cluster distances (in some unspecified metric and space) low and inter-cluster distances high. A constrained clustering algorithm optimizes its original distance criteria, as well as its conformance to the added constraints.[1]

Linkage constraints of the above form have been thoroughly investigated and shown to improve results in different application areas such as GPS lane finding [1], video and image indexing [2, 3], robot navigation [4], image segmentation [5] and text categorization [6, 7]. They are considered to increase cluster purity, decrease convergence time, and reduce error [8].

Constraints can be acquired either directly from human feedback [6], or inherently from the data collection process using explicit domain knowledge. For example, spatial or temporal proximity of data points may be encoded as a constraint [1, 2]. Another popular approach, often used for evaluation purposes, is to draw random pairs out of a labeled test set, and generate ML or CL constraints, according to the ground truth. Common to all previous work is the assumption that the constraints are consistent with some (unknown) ground truth, and contain no errors.

In contrast, this paper explores the issue of constraint noise. We take the approach, originated in this context by Davidson and Ravi [4], of treating the constraints as edges on a graph with the data points as nodes. We define intuitive noise models that may insert noise in a practical constraint acquisition scenario, and translate their effect to the generated graph. We then employ results from random graph theory to analytically investigate the effect of noise on finite-sized data sets and realistic data partitions and noise levels. We derive a quantification of the effect noisy edges may have on *any* constrained clustering algorithm, under a set of commonly-used assumptions. We also provide an experimental evaluation that shows the effect of such noise in some common UCI data sets. In particular we show that sampling an essentially linear (in the number of data points) number of edges leads to a completely useless set of constraints in common situations.

A second contribution of this paper is two fast randomized algorithms to estimate the number of classes from the graph's edges. This scenario was first laid out

*IBM Haifa Research Lab, Haifa, Israel

[1] Hard and soft variants both exist.

in [9], who observed that even finding the connected components induced by the ML edges is impractical for large data sets. Our algorithms use a very small number of edge samplings and are very efficient. They can also be made noise resistant with a little extra effort.

Paper organization. In Section 2 we define and theoretically analyze three realistic noise models. In Section 3 we present our two algorithms for estimating the number of clusters from constraints. In Section 4 we show experimental results. In Section 5 we provide concluding remarks.

2 Models of noise

In this section we define three models of noisy data, in each case analyzing the impact of noise on the quality of clustering. We view noised clustering as a two-step process. First, some noise is introduced, which distorts the ground truth constraint graph (changing node labels and/or inverting edge types). Then the clustering algorithm takes over, but is only permitted to observe this distorted ground truth rather than the actual ground truth. This two-step conceptualization allows us to decouple the effect of noise from the inner workings of the clustering algorithm—the effects we discuss are derived directly from the expected properties of the noised graph, and so are independent of the particular algorithm used. For example, the impact of noise might be a connected ML graph. Recall that a commonly used heuristic is to add the transitive closure of the ML constraints. That is, if the input includes the constraints $ML(a, b)$ and $ML(b, c)$, then the transitive constraint $ML(a, c)$ is added in pre-processing. If the noised ML graph is connected, the transitive closure is a clique, which renders *all* ML constraints equally useless, and the clustering algorithm falls back into the realm of unconstrained clustering. It may still produce good results, depending on the particulars of the data and its own implicit bias, but they will not be due to the ML constraints.

In the sequel we denote by n the number of data points which form the nodes of the constraint graph, and by k the number of classes, or clusters. Each edge is classified as either ML or CL. Our main focus will be the ML graph, i.e., the subgraph containing only the ML edges. This graph (in its undistorted form) consists simply of k cliques. We denote the error probability (reflecting the noise) by ϵ, and assume that ϵ is small (in particular, $\epsilon < 1/2$).

2.1 Noisy edges

The *noisy edges* model is that every edge, independently, inverts its type (from ML to CL or vice versa) with probability ϵ. The clustering algorithm then reads edge types directly (rather than inferring them from the node classes). In the real-world parallel of this error model constraints are generated from human feedback, as in [6]. The human operator may err, reporting an ML or a CL edge where the inverse is required. It turns out that the impact of even small noise levels is quite dramatic. We quantify this as follows.

1. If $\epsilon = \Omega(\ln n/n)$ then with very high probability the distorted ML graph is connected.
2. If $\epsilon = \Omega(\ln k/k)$ and the algorithm samples $\Omega(n)$ edges, then, assuming the clusters are equal sized,[2] the ML constraints arising from the sampled edges are sufficient (with high probability) to connect all clusters.
3. If $\epsilon = \Theta(\ln k/k)$ and the algorithm samples $\Omega(kn \ln n)$ edges, then, assuming the clusters are equal sized, the ML constraints induce (with high probability) a connected graph.

We remark that although one may question the immediate applicability of the above negative theoretical results in the real world, the empirical results observed in our experiments (see Section 4) establish their practical implications.

The wisdom emanating from these results is that this is a case where less is truly more. Unless the edge-class identification can be done very accurately, it is advisable to sample only a small number of edges. An excess of sampled edges is very likely to impair the clustering rather than assist it.

2.1.1 A word on random graphs

To prove our theoretical results we appeal to the theory of random graphs, which we now introduce briefly. A *random graph* is simply a graph formed as a product of some stochastic process. The systematic study of random graphs originated in a series of papers by Erdős and Rényi starting in the late 1950s, and rapidly bloomed into a full fledged subfield of Combinatorics. (Two good references are the books by Bollobás [10] and Janson, Łuczak, and Ruciński [11].) A significant part of the theory is devoted to the study of the manner in which graph properties (such as connectedness, chromatic number, etc.) evolve as the stochastic process progresses. For this to be meaningful, it is necessary to examine specific stochastic processes, and, indeed, several such processes have been studied extensively. One such process, which is of interest to us here, is the $G(n, p)$ *process:* it is parametrized by n and p, and is defined as follows.

[2] The *equal sized* requirement is soft. The effect lessens as the size disparity grows. The same is true for Item 3.

First, n nodes are created. Then, for every unordered pair of nodes, independently, an edge is created between these two nodes with probability p. Another random process we shall be using is the $G(n,m)$ *process.* Here a uniformly distributed set of m edges is formed between the n nodes. While the two processes may seem equivalent, they differ in the resultant edge distribution. (For example, in $G(n,p)$ edges are independent whereas in $G(n,m)$ they are not.)

The two main results that concern us are:

> In $G(n,p)$, if $p = \Omega(\frac{\ln n}{n})$, then the resultant graph is almost surely connected.
>
> In $G(n,m)$, if $m = \Omega(n \ln n)$, then the resultant graph is almost surely connected.

By *almost surely* we mean that asymptotically as $n \to \infty$, the probability of the event approaches 1. Although a lower bound on the probability of connectedness as a function of n and the constant implied in the Ω notation can be extracted from the proof of the above results, it is not very informative from a practical point of view. Suffice it to say that convergence to 1 is very rapid and the probability is very high even for moderate values of n and a small constant in the Ω term. Numerical estimates can be easily obtained by experimentation. (For example, in an experiment involving the $G(n,p)$ process with $n = 1000$ and $p = 0.01$, 99 out of 100 random graphs were connected. In a second experiment using $p = 0.02$ all 100 graphs were connected.)

2.1.2 The effect on the ground truth ML graph

THEOREM 2.1. *If $\epsilon = \Omega(\ln n/n)$ then with very high probability the distorted ML graph is connected.*

Proof. Consider the ground truth undergoing noisy-edge distortion. For every pair of nodes (independently), if they are connected by an ML edge, they will remain connected by such an edge with probability $1 - \epsilon > \epsilon$ (we are assuming ϵ is small, certainly less than 1/2). If they are connected by a CL edge, this edge will turn into an ML edge with probability ϵ. Thus, referring to the observable must-link graph, we see that it is actually a random graph created by a $G(n,p)$ process (with $p = \epsilon$), and it is therefore connected with high probability.[3]

The import of this is quite clear: *for even moderate n and fairly small ϵ the ML graph seen by the clustering algorithm is connected with very high probability.* Note that this is true regardless of structure of the undistorted ML graph. (The exact probability of becoming connected will obviously depend on the original must-link structure, but in all cases it will be very close to 1.) Thus the ML graph almost surely conveys no useful information.

2.1.3 The effect on cluster separation

Although, as we have just seen, the full ML graph is useless, a clustering algorithm can still arguably make use of it, since such an algorithm will typically sample only a relatively small number of edges, and most of these edges will be reported correctly, so the algorithm will not discover that the graph is connected. It is therefore necessary for us to delve deeper in order to understand the actual impact of errors.

We analyze the case where the clusters are equal-sized and the sampling process is repeated uniformly distributed i.i.d. node-pair sampling. (In other words, *sampling m edges* means repeating m times independently: select a random, uniformly distributed, node-pair.) Let us define the *observed graph* as the subgraph of the constraint graph in which only the edges sampled by the algorithm and reported as ML are retained, and let us also define the *observed cluster graph* as the graph obtained from the observed graph by shrinking each true cluster (i.e., the nodes belonging to the same cluster in the undistorted ground truth) to a single node and placing an edge between two nodes if the observed graph contains an edge connecting any two nodes in the corresponding clusters.

THEOREM 2.2. *If $\epsilon = \Omega(\ln k/k)$ and the algorithm samples $\Omega(n)$ edges, then with high probability the observed cluster graph is connected.*

Proof. Let us view the random process which constructs the graph as a two-step process in which the noise is added last. Specifically, start by constructing an observed cluster graph from the actual ground truth without applying any noise. Call this graph the *potential graph.* Then apply noise to the ground truth, and obtain the true observed cluster graph from the potential graph as follows. For every edge (u,v) in the potential graph, retain this edge if and only if, among the edges connecting the clusters corresponding to u and v, at least one of the edges sampled in the first step has become an ML edge due to the noise. Note that (given a particular outcome of the first step) the probability of an edge surviving through the second step is at least ϵ, and that these events (i.e., survival of edges) are independent.

Now consider a particular pair of nodes (u,v) in

[3] Strictly speaking, the process is not quite $G(n,p)$ because some of the pairs—the ones connected by ML edges in the actual ground truth—have probability greater than ϵ of becoming connected by an ML edge. But this can only hasten the onset of connectedness.

the potential graph. There are $\left(\frac{n}{k}\right)^2$ edges running between the two corresponding clusters in the ground truth ML graph. Suppose the clustering algorithm samples $\Omega(n)$ edges—for simplicity and concreteness let us assume exactly n edges. The probability that a particular sampling step will pick an edge in this group is roughly (using the approximation $\binom{n}{2} \approx n^2/2$) $\frac{(n/k)^2}{n^2/2} = \frac{2}{k^2}$. Thus the probability that (u, v) will *not* appear as an edge in the potential graph is at most (roughly) $(1-2/k^2)^n \approx e^{-2n/k^2}$. Using the union bound we get that the probability that the potential graph is not fully connected is at most $\frac{k^2}{2}e^{-2n/k^2}$, which is negligible for reasonable values of n and k. (E.g., for $n = 500$ and $k = 10$ it evaluates to 0.0023.) Thus the potential graph is fully connected with high probability.

Given that the potential graph is indeed fully connected, the second step, in which noise is applied, can be viewed as a $G(n, p)$ process on the k nodes of the potential graph with $p = \epsilon$. Thus for error probability $\epsilon = \Omega(\frac{\ln k}{k})$ the observed cluster graph will be connected with high probability. Asymptotically, then, for large k and significantly larger n ($n \gg k^2$), even minuscule error probabilities almost guarantee that the observed cluster graph will be connected.

The edges of the observed cluster graph represent the inability of the algorithm to distinguish between different clusters. Assuming the clustering algorithm is fairly good at identifying the intra-cluster relations (based on metric distances, etc.), the added ML constraints force it to fuse clusters that should actually remain distinct. The worst case scenario, in this sense, is when the observed cluster graph is connected, implying a single cluster. If, on the other hand, the algorithm fails to identify the intra-cluster relations accurately, then the situation is arguably even worse: not only are clusters being spuriously split up, but bits of distinct clusters are being glued together. In either case, the constraints are actually worse than useless. This is because too many are being used.

2.1.4 The effect on the connectedness of the observed ML graph Our final result examines the situation when the number of sampled edges is increased from $\Omega(n)$ to $\Omega(kn \ln n)$ (and the clusters are equal sized). We show that the observed graph becomes connected with high probability.

THEOREM 2.3. *If $\epsilon = \Theta(\ln k/k)$ and the algorithm samples $\Omega(kn \ln n)$ edges, then with high probability the observed graph is connected (for sufficiently large n and k, and $n \gg k^2$).*

Proof. To ease the exposition we break the proof into five steps. We denote by l the number of edges sampled by the clustering algorithm, and by r the number of edges within a cluster in the ground truth. Since the clusters are equal sized, we have $r = \binom{n/k}{2} \approx n^2/(2k^2)$. We also approximate the total number of edges by $\binom{n}{2} \approx n^2/2$. The next four lemmas refer to the subgraph of the ground truth induced by an arbitrary cluster. Following these lemmas, we wrap everything up in a final proof of the theorem.

LEMMA 2.1. *The probability that more than half of the edges in the subgraph vanish from the distorted ML graph is negligibly small compared with $1/k$.*

Proof. The probability that a particular set of $r/2$ edges vanish is $\epsilon^{r/2}$. There are $\binom{r}{r/2} \le (\frac{r \cdot e}{r/2})^{r/2} = (2e)^{r/2}$ such sets, so applying the union bound we can bound the probability that at least $r/2$ edges vanish by $(2e)^{r/2}\epsilon^{r/2}$, which is negligibly small since $\epsilon = \Theta(\ln k/k)$ and we are assuming large k.

LEMMA 2.2. *Given that the set of surviving edges in the subgraph contains at least $r/2$ edges, the probability that the sampling process hits this set less than $lr/(2n^2)$ times is negligibly small compared with $1/k$.*

Proof. The probability of a single sampling step hitting the set of surviving edges is at least $(r/2)/(n^2/2) = r/n^2$. Denoting by X the number of hits, we therefore have $E[X] \ge lr/n^2$. Thus $\Pr\left(X < lr/(2n^2)\right) \le \Pr(X < \frac{1}{2}E[X])$. Since the sampling steps are independent, we can apply Chernoff's bound $\Pr(X < (1-\delta)E[X]) < e^{-\frac{1}{2}E[X]\delta^2}$ with $\delta = 1/2$ to obtain $\Pr(X < lr/(2n^2)) < e^{-lr/(8n^2)}$. Plugging in $l = \Omega(nk \ln n)$ and $r \approx n^2/(2k^2)$, together with $k^2 \ll n$ gives the desired result.

LEMMA 2.3. *Given that the set of surviving edges in the subgraph contains at least $r/2$ edges and that the sampling process hits this set at least $lr/(2n^2)$ times, the probability that it hits less than $lr/(6n^2)$ distinct edges in this set is negligibly small compared with $1/k$.*

Proof. Let $t \ge r/2$ be the number of surviving edges, and let us focus on the first $lr/(2n^2)$ hits. Consider a particular surviving edge. Since the hits are independent and uniformly distributed, the probability that the edge is hit in a single sampling step (among the first $lr/(2n^2)$ hits) is $1/t$, and therefore the probability that it is hit more than three times is less than

$$\begin{aligned} & \textstyle\sum_{i=4}^{lr/(2n^2)} \binom{lr/(2n^2)}{i} t^{-i} \\ \le\ & \textstyle\sum_{i=4}^{lr/(2n^2)} \left(\frac{lr}{2n^2 t}\right)^i \\ \le\ & \textstyle\sum_{i=4}^{lr/(2n^2)} (l/n^2)^i \\ <\ & 2(l/n^2)^4, \end{aligned}$$

where the last inequality follows from $l \ll n^2$. Thus the probability that at least one of the edges is hit more than three times is (by the union bound) less than $2r(l/n^2)^4$, which is negligibly small since $l = \mathrm{O}(kn \ln n)$ and $r \approx n^2/(2k^2)$ and we are assuming large k and n, and $k^2 \ll n$. If there are at least $lr/(2n^2)$ hits, and no edge is hit more than three times, the number of distinct edges hit must be at least $lr/(6n^2)$.

Lemma 2.4. *Given that at least $lr/(6n^2)$ distinct edges have been sampled and identified correctly as ML edges, the probability that the corresponding cluster is not connected in the observed ML graph is negligibly small compared with $1/k$.*

Proof. The set of sampled edges (within the subgraph) is uniformly distributed (since the sampling procedures favors no particular set of edges). Therefore the subgraph of the observed ML graph corresponding to the cluster is the product of a $G(n, m)$ process on the n/k nodes of the cluster, with $m \geq lr/(6n^2) = \Omega(\frac{n^2/(2k^2)\cdot kn \ln n}{6n^2}) = \Omega(\frac{n}{k} \ln n) = \Omega(\frac{n}{k} \ln \frac{n}{k})$. Thus it is almost surely connected.

Proof. [Proof of Theorem 2.3] We have already seen that $\Omega(\ln k/k)$ edges are sufficient for the observed cluster graph to be connected. It thus suffices to show that when $\Omega(kn \ln n)$ edges are sampled, each of the clusters becomes connected (with high probability) in the observed ML graph (since the combination of these two events implies that the entire observed ML graph is connected). The probability that a given cluster is not connected is bounded by the sum of the four negligibly small probabilities explored in the previous four lemmas. Since each of this probabilities is negligible relative to $1/k$, applying the union bound on the k clusters gives the desired result.

2.2 Noisy nodes

In the *noisy nodes* model the errors occur as follows. For each node, independently, the node is assigned a new class (say, chosen uniformly from the set of all classes excluding the node's original class) with probability ϵ, and is left undisturbed with probability $1-\epsilon$. The motivation for this error model is a scenario where there is some inherent error in the low-level observation process causing instances to be misidentified. We assume that constraints are generated in pre-processing, after observing the instances, and are therefore consistent with the observed labels. This kind of constraint generation is standard practice in experimentation with UCI data.

The noisy nodes model is quite robust in the sense that the effect of noise is simply to distort the ground truth by changing the class of a small number of nodes. The impact this may have on a clustering algorithm will depend on the interplay between the actual ground truth and the algorithm in question. A good clustering algorithm can be expected to suffer only slightly from this distortion.

2.3 Inconsistent noisy nodes

The *inconsistent noisy nodes* model is something of a hybrid between noisy nodes and noisy edges. Specifically, for each pair of nodes (independently), the pair is examined, and depending on whether both are seen to belong to the same class, the edge between them is classified as ML or CL. When a node is examined, the class returned is correct with probability $1 - \epsilon$ and incorrect with probability ϵ. In the latter case the class returned is uniformly distributed in the set of all classes excluding the node's true class. The key difference between this model and the plain noisy nodes model is that the outcome of a given node examination is independent of any other node examinations, *including examinations of the same node,* i.e., the same node may report different classes when queried multiple times. The motivation for this error model comes from an observation system that simultaneously generates the instances and the associated constraints. It does so automatically from background knowledge such as spatial or temporal proximity. This is the model used in [1], but we assume a fully-automatic mode where errors are introduced (e.g., in marking of the lane shifts, or in the identity of the vehicle).

True to its hybrid nature, the behavior of this model is somewhere between that of noisy nodes and noisy edges. To see this, observe that a CL edge is misclassified with probability $\epsilon^2/(k-1)$ (both nodes report identical incorrect classes). An ML edge is misclassified with probability $2\epsilon(1-\epsilon)+\epsilon^2(1-1/(k-1))$ (either exactly one node reports an incorrect class, or both nodes report distinct incorrect classes). In effect, we get the noisy edges model, but with a smaller error probability for the CL edges and a nearly doubled error probability for the ML edges. The remainder of the analysis is similar to the analysis of the noisy edges model, with suitably modified error parameters.

3 Computing the number of clusters

One of the obstacles to accurate clustering is often the lack of knowledge of the "true" number of clusters. Indeed, many clustering algorithms require this number to be fixed apriori as part of the input. While there does not always exist a "correct" answer, the underlying assumption in constrained clustering is that the ground truth does indeed partition the data points into a well defined set of clusters, which can be easily identified by examining all of the graph edges. However, as pointed out by Davidson and Ravi [9], this may be impractical when the size of the input is large.

In this section we propose two fast randomized algorithms to compute the number of clusters with high probability by sampling a small number of edges. The first algorithm, named **NumClustersThresh**, makes the natural assumption that we are mainly interested in sufficiently large clusters, namely, clusters of size at least some fraction p of the overall graph size. With probability at least $1-p$ the algorithm returns a number between the number of such clusters and the number of all clusters, requiring $O((\frac{1}{p}\ln\frac{1}{p})^2)$ edge samplings, which is quite attractive for values of p arising in practice (e.g., $p = 0.05$). The error probability can be decreased exponentially by re-running several times and taking the maximum result.

```
Algorithm NumClustersThresh($(G, p)$)
Input: Constraint graph $G$ on $n$ nodes; fraction $p$.
Output: Number of big clusters, i.e., clusters of
        size at least $pn$.
1.  Choose independently and uniformly $r = \frac{2}{p}\ln\frac{1}{p}$
    nodes $U = \{u_1, \ldots, u_r\}$.
2.  For each index pair $1 \le i < j \le r$ do:
3.      If $(u_i, u_j)$ is an ML edge,
4.          $U \leftarrow U \setminus \{u_i\}$.
5.  Return $|U|$.
```

Theorem 3.1. *The number returned by Algorithm **NumClustersThresh** does not exceed the number of clusters in G, and is at least the number of big clusters with probability at least* $1 - p$.

Proof. We say that a cluster is *represented* in U if U contains a node from that cluster. Let $\mathcal{C}$ denote the set of clusters initially represented in U. Since every two nodes coming from the same cluster are connected by an ML edge, and every two nodes coming from different clusters are not connected by an ML edge, the *For each* loop leaves in U exactly one node from each cluster in $\mathcal{C}$. Thus the number returned by the algorithm is precisely $|\mathcal{C}|$, which is at most the number of clusters in G. To complete the proof we need only show that $\mathcal{C}$ contains all of the big clusters with probability at least $1-p$. Let C be a big cluster. The probability of not sampling a node from C in a single sampling step is at most $1 - p$. Therefore, the probability that C is not represented in U is bounded by $(1-p)^r = (1-p)^{2/p \ln 1/p} < p^2$. Since the number of big clusters is at most $1/p$, applying the union bound gives the desired result.

Observation 3.1. *The number of edge samplings performed by Algorithm **NumClustersThresh**, and its running time, are* $O((\frac{1}{p}\ln\frac{1}{p})^2)$.

The drawback of **NumClustersThresh** is that the number of edge samplings it requires depends on the potential maximum number of clusters of interest (i.e., $1/p$), which may be substantially greater than the actual number of such clusters. Our second algorithm, named **NumClusteresBalanced**, partially redresses this deficiency by replacing the "absolute" threshold p with a relative one. Specifically, it assumes that we are mainly interested in clusters of size at least some fraction α of the size of the largest cluster. The algorithm outputs a number which is between the number of such clusters and the number of all clusters with probability at least 3/4. The number of edge samplings is $O((l/\alpha)^2 \ln^2(l/\alpha))$ where l is the number of clusters detected by the algorithm. Once again, the error probability can be decreased exponentially by re-running the algorithm.

```
Algorithm NumberClustersBalanced$(G, \alpha)$
Input: Constraint graph $G$ on $n$ nodes; fraction $\alpha$.
Output: Number of clusters in G.
1.  Initialize: $c \leftarrow 4$; $t \leftarrow 4$; $\alpha \leftarrow \min\{\alpha, 1/2\}$.
2.  While $c \ge t$ do:
3.      $t \leftarrow \max\{2t, c\}$.
4.      $c \leftarrow$ NumClustersThresh$(G, \frac{\alpha}{t})$.
5.  Return $c$.
```

Theorem 3.2.

1. *The number returned by Algorithm **NumClusters-Balanced** is computed by Algorithm **NumClustersThresh**, so it cannot exceed the number of clusters in G (by Theorem 3.1).*

2. *If the size ratio between the smallest and largest clusters is at least α, the algorithm returns the number of clusters with probability at least* 3/4.

3. *The number of edge samplings performed by the algorithm, and its running time, are $O((l/\alpha)^2 \ln^2(l/\alpha))$, where l is the number of clusters detected by the algorithm.*

Proof. Let k be the number of clusters in the graph. In each iteration the algorithm tests whether the graph contains at least t clusters by calling Algorithm **NumClustersThresh**. Assuming that the size threshold α/t is sufficiently low for Algorithm **NumClustersThresh** to return a number that is at least t as long as $t \le k$, the loop will continue until t surpasses k. Once t surpasses k, the algorithm will halt since the number returned by Algorithm **NumClustersThresh** is at most k (the true number of clusters), which is less than t. This will occur after (at most) a logarithmic number of iterations, since t (at least) doubles each iteration. The graph contains k clusters, so the size of the largest is at least n/k, and therefore the size of the smallest is at least $\alpha n/k$. In the last iteration $t > k$, so the threshold α/t is sufficiently small for Algorithm **NumClustersThresh** to succeed (i.e., return k) with probability at least $1 - \alpha/t$ (by Theorem 3.1). To bound the overall failure probability of Algorithm **NumClustersBalanced**, let us consider the failure probability at the ith iteration, conditioned on the event that the iteration occurs. We denote by t_i the value of t used in the call to Algorithm **NumClustersThresh** in the ith iteration. In the iteration in which $t_i > k$ we have already seen that the failure probability is at most α/t_i, and this iteration is last. In the preceding iterations $t_i \leq k$, and the failure probability is the probability that Algorithm **NumClustersThresh** will nonetheless return a number smaller than t_i. The threshold used is α/t_i, which is sufficient for Algorithm **NumClustersThresh** to detect at least t_i clusters[4] with error probability at most α/t_i. By the union bound, the overall failure probability of Algorithm **NumClustersBalanced** is bounded by the sum of the above conditional probabilities: $\alpha \sum_i \frac{1}{t_i}$. Since $t_1 = 8$ and $t_{i+1} \geq 2t_i$, we can bound the above sum by $\frac{1}{8}\alpha \sum_{i=0}^{\infty} 2^{-i} < \frac{1}{4}$.

It remains to bound the number of edge samplings. Summing over all iterations, the total number of edge samplings is $\mathrm{O}(\sum_i (t_i/\alpha)^2 \ln^2(t_i/\alpha))$. Since t ranges from 8 to at most $2l$, where l is the number of clusters detected by the algorithm, we get a bound of $\mathrm{O}((l/\alpha)^2 \ln^2(l/\alpha))$.

Theorem 3.2 tells us that if the imbalance parameter α is set correctly, Algorithm **NumClustersBalanced** will find the correct answer using a number of edge samplings that is (roughly) quadratic in k/α. If, however, α is chosen larger than the actual smallest to largest cluster size ratio, the performance of the algorithm may vary between two extremes. The good case is when most of the nodes are contained in clusters whose sizes are within α of the largest. In this case the algorithm will tend to count only these large clusters, but will return a higher number (and run longer) with some relatively small probability. The bad case is when the majority of nodes are contained in very many, very small clusters (at the extreme, single node clusters). Essentially, the data is unclustered. In this case the algorithm will tend to detect many of the small clusters and perform many edge samplings. (Note, however, that the algorithm never overestimates the true number of clusters.) In Section 4 we provide some experimental results concerning these effects.

3.1 Dealing with noisy data Turning to the case of noisy data (either noisy edges or inconsistent noisy nodes), there are two problems. The first is that ML edges may be reported as CL, leading Algorithm **NumClustersThresh** to retain more than one node per cluster. We can overcome this difficulty easily by counting the connected components induced by the edges between all sampled node pairs. Assuming each cluster is represented by several sampled nodes (which may require a moderate increase in the number of nodes to sample), the probability that the clique induced by these nodes will become disconnected due to noise is very small, so we can simply count connected components. The second problem is CL edges reported as ML, causing distinct connected components to fuse. We can overcome this by cutting up connected components based on minimum cuts. Because the noise probability is small (and again assuming clusters are reasonably represented in U), the number of edges in the cut separating two clusters will be much smaller than in cuts within the cluster. We can therefore look for the minimum cut inside each connected component, and if it is below a reasonable threshold (defined relative to the number of nodes in the component) we discard these edges and repeat.

There is a contrast between these techniques and Section 2. On the one hand, identifying the classes correctly from constraints in the presence of even minor noise is impractical. On the other hand, getting their number right can be done very efficiently. In addition to this shift of focus, Section 2 deals with the case of unstructured, randomly chosen edges, whereas here we actively sample edges that (noise notwithstanding) induce separate cliques.

4 Experiments

In this section we make the theoretical discussion of Section 2 more concrete by providing simulation results of different noise scenarios on commonly-used data sets. We also present experimental results concerning the behavior of Algorithm **NumClustersBalanced** from Section 3.

4.1 Effect of Noise on Clustering. We experimented with several UCI data sets (see Figure 1). The experimental setup was as follows. The UCI data was stripped of all metric information, leaving just the iden-

[4] Actually, it is sufficient to detect t_i clusters assuming the graph contains exactly t_i clusters. Intuitively, it seems obvious that if, as is our case, the graph contains more that t_i clusters (all obeying the α parameter), its probability of detecting at least t_i clusters can only be greater. There is a straightforward proof of this for the case $\alpha \leq 1/2$. This is the reason for the initialization line: $\alpha \leftarrow \min\{\alpha, 1/2\}$.

tity of the points and their labels. A random pair was drawn from this data, noised according to the particular noise model, and output as a constraint. This step was repeated as necessary. Note that the number of edges of a particular type (ML or CL) is only known in expectation. For example, on a balanced two-class set, we expect half the generated edges to be ML, as compared to a quarter in a four-class balanced set. In the graphs below, the X axis shows the total number of edges drawn, as a fraction of n (the number of nodes). Note that n may vary from one dataset to another. The noise rate ϵ was set to 1%, and experiments were repeated at least 100 times. Error bars are shown, but are generally too small to be seen clearly.

We first explored the effect of noisy ML edges after applying transitive-closure processing. For each set of noised ML edges drawn as above we computed the number of connected components in the resulting graph. Figure 2(a) shows the probability that there is just one connected component. We see that it is close to 0 as long as the number of edges is up to $5n$. The transition to connectedness happens fast, as expected, and is generally complete at $10n$. The exceptions are the datasets with more than 3 classes, namely glass and e-coli. Since for most algorithms and usage scenarios the number of constraints is very small, they are generally on the safe side under this metric.

However, the experiment above is unrealistic in at least two respects. First, it does not deal with the case where some components join, forming spurious links, but not enough to connect the whole graph. Second, the metric data in the input is often used to cluster the data points together, whereas the experiment ignores this. The second experiment addresses this second point (discussed theoretically in Theorem 2.2). The edges were drawn as before, but prior to transitive closure augmentation, intra-class edges were added (so that each input class was guaranteed to form a clique). The results are shown in Figure 2(b). Here, the collapse of the constraint graph occurs much earlier. For two-class inputs, which are common, as few as $0.2n$ edges (ML and CL combined) are enough to trigger it with probability greater than 1/2. Other inputs seem to be on safer ground, but they too are affected by spurious links, as shown in Figures 3–5. Figure 3 shows the number of connected components as a function of the number of edges sampled in the noisy edges model; Figure 4 shows the probability that the graph is not connected as a function of the number of edges sampled in the inconsistent noisy nodes model; and Figure 5 shows the number of connected components as a function of the number of edges sampled in the inconsistent noisy nodes model.

4.2 Behavior of Algorithm NumClustersBalanced.

Initially, we ran Algorithm **NumClustersBalanced** on the same UCI data sets, but because these data sets are small, the algorithm did not exhibit any interesting behavior. In all cases it identified the true number of clusters in up to 3 iterations, even when using relatively large values of α.

In order to gain some insight into the algorithm's behavior on larger data sets, especially ones with different mixes of cluster sizes, we ran it on three synthetically generated data sets, each consisting of approximately 1000 nodes and 15–20 clusters. The first data set, labeled A1, consisted of 19 clusters whose sizes formed an arithmetical progression: $5, 10, 15, \ldots, 95$. The second data set, labeled A2, consisted of 10 clusters of size 100 and 5 clusters of size 5. The last data set, A3, consisted of one cluster of size 500 and 20 clusters of size 25. For each data set, the algorithm was run 100 times. No noise was introduced in these experiments.

Figure 6(a) shows the average fraction of edges that were sampled by the algorithm as a function of α, and Figure 6(b) shows the average number of clusters detected by the algorithm as a function of α. In both graphs α is given as a percentage. (The graphs end at $\alpha = 50\%$ because the algorithm resets $\alpha > 1/2$ to $\alpha = 1/2$.) We see that in all cases the number of edges sampled is exceedingly small (and negligibly small relative to our theoretically derived bound). We also see that on A1 and A3, the algorithm successfully identified the correct number of clusters and was completely insensitive to α. On A2 it performed very well even for α as large as 1/5 (the correct value for all inputs is $\alpha = 1/20$), after which its performance declined roughly linearly. These results indicate that our algorithm is fast, accurate, and robust.

5 Conclusion

Although theoretically flavored, this is also a practical paper in the sense that practitioners can directly apply lessons learned from it. We ground our results in theory, but try to frame them in a realistic setting with finite quantities. We give an analysis of the onset of connectedness in the presence of constraint noise, and suggest randomized algorithms for estimating the number of clusters. We also provide experimental results demonstrating the actual phenomena.

There is probably much more that can be said on the topic. We stop short of listing techniques of dealing with noise. These can be in the form of general guidelines (other than advising against the transitive closure), or algorithm-specific improvements as in [12]. This is an obvious area of future work. Our vision is that as pairwise constraints gain popularity, they will

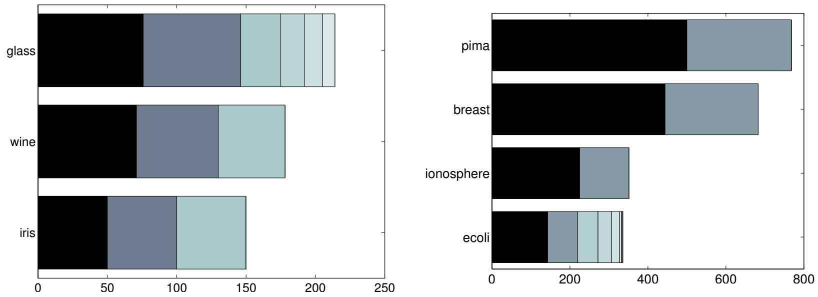
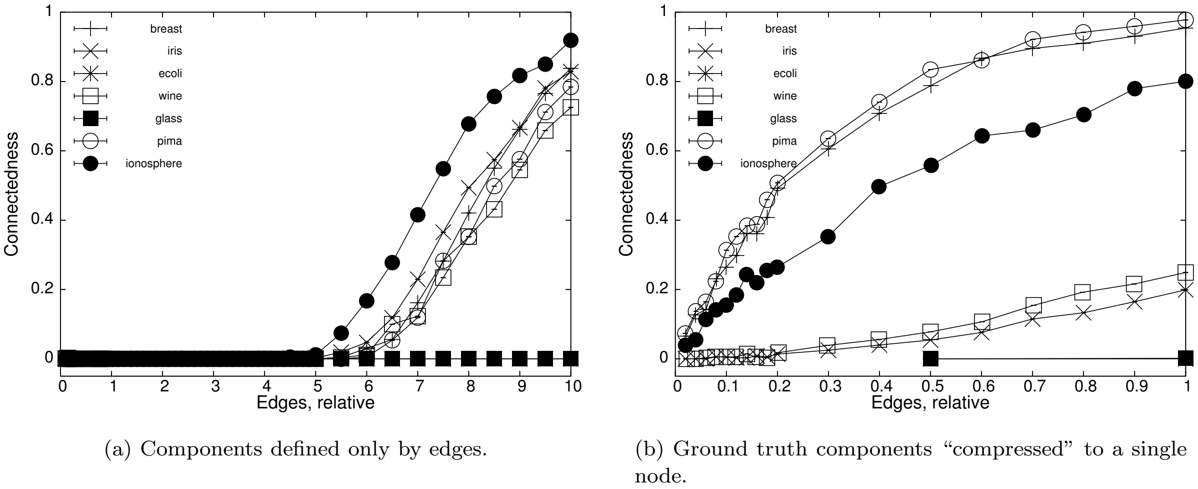


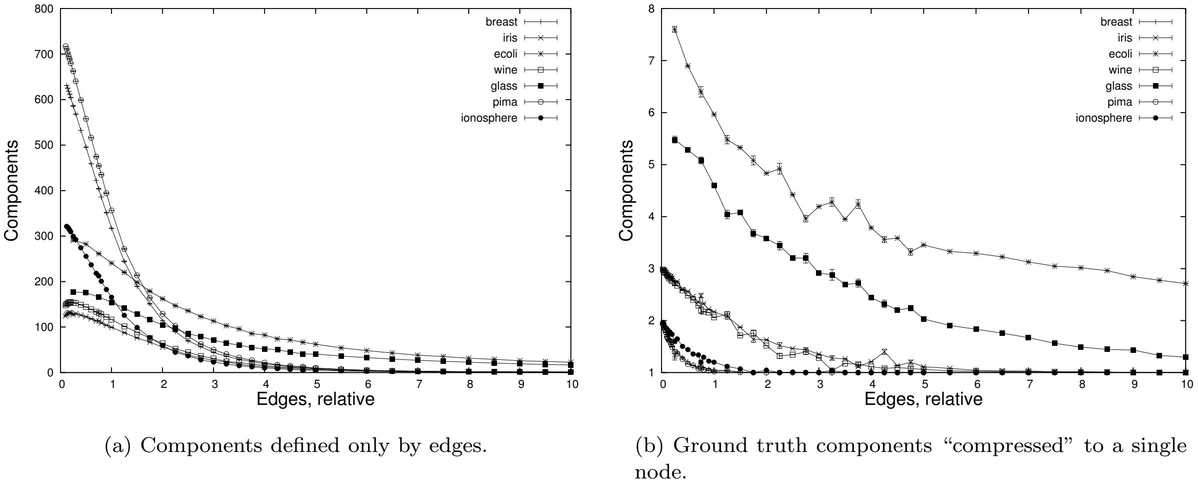
Figure 1: Sizes and class sizes of data sets used



(a) Components defined only by edges.

(b) Ground truth components "compressed" to a single node.

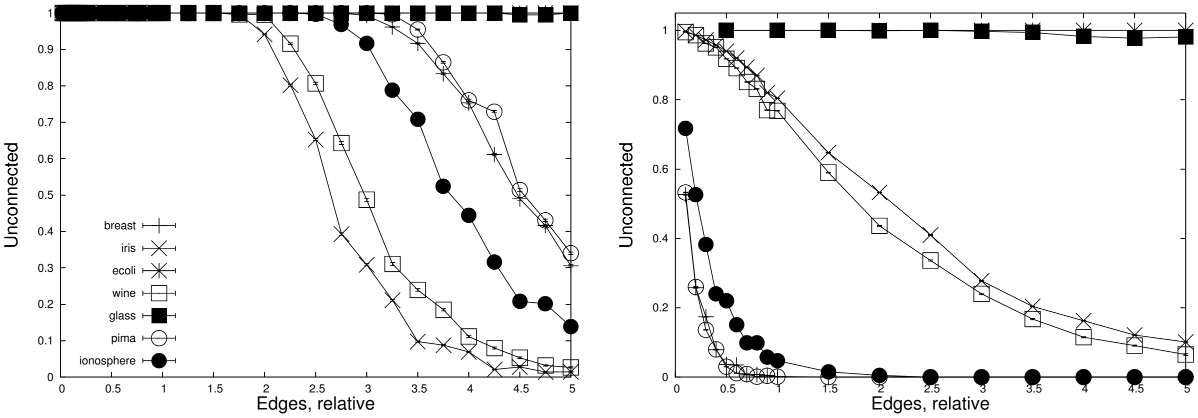
Figure 2: Probability of connectedness vs. number of drawn edges (in the noisy edges model).



(a) Components defined only by edges.

(b) Ground truth components "compressed" to a single node.

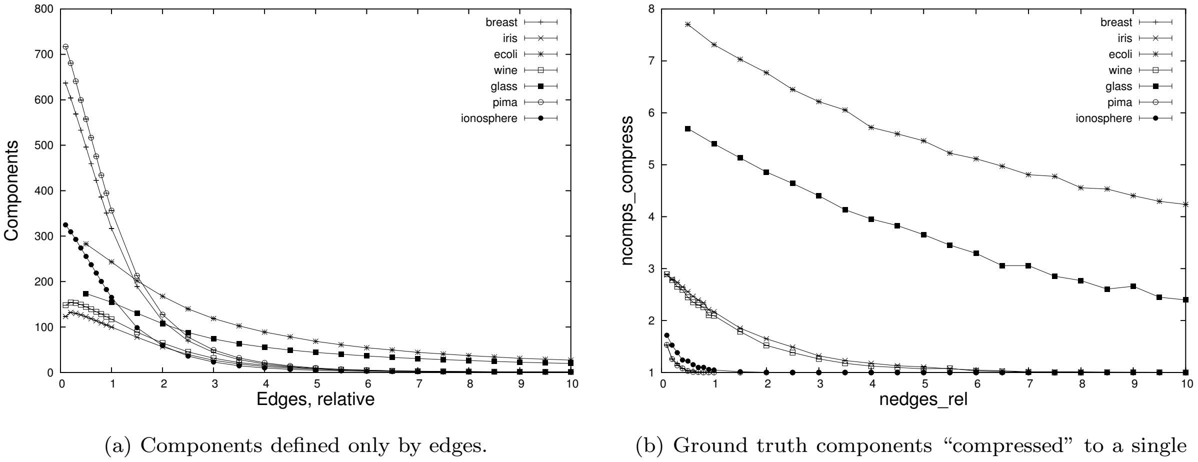
Figure 3: Number of connected components vs. number of drawn edges, under the noisy edges model.



(a) Components defined only by edges.

(b) Ground truth components "compressed" to a single node.

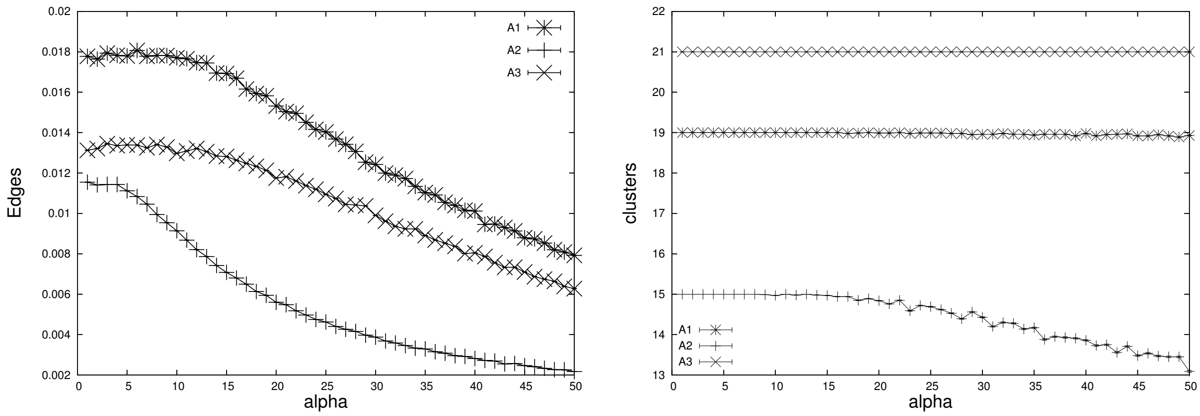
Figure 4: Probability of existence of more than one component vs. number of drawn edges, under the inconsistent noisy nodes model.



(a) Components defined only by edges.

(b) Ground truth components "compressed" to a single node.

Figure 5: Number of connected components vs. number of drawn edges, under inconsistent noisy nodes model.



(a) Fraction of edges sampled.

(b) Number of clusters detected.

Figure 6: Performance of **NumClustersBalanced** vs. α

be acquired from more diverse sources. The inevitable outcome will be introduction of various forms of noise. Surveying these, and producing matching analyses for them, is another area of future work.

References

[1] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. In Carla Brodley and Andrea Danyluk, editors, *Proceeding of the 17th International Conference on Machine Learning*, San Francisco, CA, 2001. Morgan Kaufmann.

[2] Wei-Hao Lin and Alexander Hauptmann. Structuring continuous video recordings of everyday life using time-constrained clustering. In *IS&T/SPIE Symposium on Electronic Imaging*, San Jose, CA, January 2006.

[3] T. Hertz, N. Shental, A. Bar-Hillel, and D. Weinshall. Enhancing image and video retrieval: Learning via equivalence constraints. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[4] Ian Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *5th SIAM Data Mining Conference*, 2005.

[5] Stella X. Yu and Jianbo Shi. Grouping with directed relationships. *Lecture Notes in Computer Science*, 2134, 2001.

[6] David Cohn, Rich Caruana, and Andrew McCallum. Semi-supervised clustering with user feedback. Technical report, Cornell University, 2003. TR2003-1892.

[7] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, Seattle, WA, August 2004. ACM.

[8] Sugato Basu and Ian Davidson. Clustering with constraints: Theory and practice. Online Proceedings of a KDD tutorial, 2006. `http://www.ai.sri.com/~basu/kdd-tutorial-2006/`.

[9] Ian Davidson and S. S. Ravi. Intractability and clustering with constraints. In *Proc. of International Conference on Machine Learning*, 2007. To appear.

[10] Bélla Bollobás. *Random Graphs (2nd ed.).* Cambridge University Press, 2001.

[11] Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random Graphs.* John Wiley & Sons, 2000.

[12] Blaine Nelson and Ira Cohen. Revisiting probabilistic models for clustering with pair-wise constraints. In *Proc. of International Conference on Machine Learning*, 2007. To appear.

[13] Dan Pelleg and Dorit Baras. k-means with large and noisy constraint sets. In *ECML*, 2007.