Lagrangian Constrained Clustering

Mohadeseh Ganji James Bailey Peter J. Stuckey
The University of Melbourne

Abstract

Incorporating background knowledge in clustering problems has attracted wide interest. This knowledge can be represented as pairwise instance-level constraints. Existing techniques approach satisfaction of such constraints from a soft (discretionary) perspective, yet there exist scenarios for constrained clustering where satisfying as many constraints as possible. We present a new Lagrangian Constrained Clustering framework (LCC) for clustering in the presence of pairwise constraints which gives high priority to satisfying constraints. LCC is an iterative optimization procedure which incorporates dynamic penalties for violated constraints. Experiments show that LCC can outperform existing constrained clustering algorithms in scenarios which satisfying as many constraints as possible.

Keywords Constrained clustering, Semi-supervised learning, Lagrangian multipliers method.

1 Introduction

Clustering is an important task in data mining and machine learning which has a wide variety of applications in real-world problems. Many different algorithms have been proposed for this unsupervised learning problem. However, there often exists some pre-knowledge about the true object memberships which can be incorporated into clustering algorithms to enhance the quality of the clustering. This side information can be obtained from an oracle or expert knowledge (about the domain or data set) or it might be the result of high precision pilot-experiments. In clustering problems, this information can be represented as instance-level or cluster-level constraints. Using such constraints can also result in greater algorithmic efficiency.

Due to its wide application, the problem of taking advantage of side-information in data clustering, which is called semi-supervised or constrained clustering, has been widely studied over the last decade [1, 2]. Typically, the side information is represented as pairwise instance-level constraints which model whether a pair of objects must lie within the same cluster (a *must-link (ML)* constraint) or lie within different clusters (a *cannot-link (CL)* constraint).

These types of constraints have been investigated in applications like GPS lane finding [3], image and video segmentation [4, 1] and text categorization [5]. Many existing clustering algorithms, including the k-means algorithm, have been adapted to work with constraints [3, 6]. However, to our knowledge, no existing constraint clustering algorithm aims to satisfy *all* the constraints. Rather, satisfying the constraints is desirable, but discretionary, and some algorithms fail and terminate when facing a constraint they cannot satisfy. In some other approaches, a degree of belief is defined to tackle such situations and allow a percentage of constraints to remain violated. However, these algorithms are challenged to reach a good solution when a large number of constraints need to be satisfied.

Our focus in this work is on situations where very high priority must be given to satisfying constraints. The success of an algorithm in this context is therefore judged by how many constraints it satisfies, in addition to the quality of the resulting clustering. This perspective is suitable for situations where the constraints are guaranteed to represent ground truth. We give three illustrations. Firstly, the constraints may describe a known physical or mathematical property of a system, where the user has complete confidence about the cluster membership for some objects, E.g. A pair of objects in a road network that are unreachable (cannot-link) due to a physical barrier. Secondly, the constraints may be derived using a procedure that discovers ground truth (cluster membership) for a small sample of objects, such as when using an expensive/invasive medical test like a biopsy. Thirdly, some contexts such as image segmentation allow the user to easily generate a large number of constraints having very high confidence (pairs of pixels which do/do not lie in the same segment).

Making an algorithmic commitment to satisfying (almost all) constraint brings new challenges, particularly in regard to efficiency. The approach we will present proposes a novel constrained clustering algorithm which targets to satisfy as many constraints as possible in a reasonable time. Constrained clustering is most challenging in the presence of CL constraints. The CL-feasibility problem is NP-complete as it is reducible to the Graph k-Colorability problem [7] which is known to be NP-complete [8]. To tackle this aspect, our algorithm is based on Lagrangian relaxation [9] to satisfy constraints. Lagrangian relaxation is an optimization approach for solving constrained problems. It has been successfully used for solving hard graph coloring problems [10] and it has been shown to work effectively for many classes of hard constrained optimization problems [11, 10]. We propose a new algorithm based on a Lagrangian multipliers method, which given a clustering problem with a set of must-link and cannot-link constraints, clusters the data to satisfy as many constraints as possible.

In brief, the main contributions of this paper are as follows:

- We address the problem of constrained clustering for situations where satisfying all constraints has high priority and we propose a new Lagrangian based constrained clustering algorithm (called LCC) which concentrates on satisfying constraints.
- Our proposed LCC algorithm not only achieves very high constraint satisfaction, but also achieves competitive performance in terms of quality and robustness of the clustering results.

2 Related Work

Because of increasing interest in constrained clustering, existing clustering algorithms have been adapted to incorporate constraints. The COP-$k$means algorithm [3] is one of the variations of k-means which can incorporate instance level constraints. This algorithm, in a first step, creates super instances based on the transitive closure of must-link constraints, and in a second step, tries to assign each super instance to its closest feasible cluster center. Although COP-$k$means can incorporate constraints, its greedy approach and inability to change cluster assignments once they have been made makes it vulnerable to the number and distribution of constraints. It may reach a dead end and be unable to find any feasible assignment and stop while the problem is feasible.

Later research has focused on cases where inconsistency in constraints is a concern. Therefore, algorithms have been developed to be able to work with noisy or inconsistent constraints by letting some of them remain unsatisfied if it is impossible or too difficult to satisfy them. To this end, Basu et al [12] introduced the PCKmeans algorithm which allows constraints to remain unsatisfied if the cost of satisfying them is high or the problem is infeasible. PCKmeans uses the instance-level constraints in an initialization step to find a good estimation of initial centres to start with. In this framework, the must-link set is augmented by transitive closure and the cannot-link set becomes augmented by adding cannot-link constraints for neighbours of initial pairs which are constrained by a cannot-link constraint.

A more flexible k-means style constrained clustering algorithm called CVQE [6] allows constraints to remain unsatisfied if the cost of satisfying them is high or the problem is infeasible. This algorithm has an objective function including penalty terms for violated constraints. Each assignment of constraint instances is considered and upon violation of constraints, a penalty is added to the objective which is based on the distance between cluster centres. Checking all possible assignments for cannot-link constraints requires $O(k^2)$ comparisons.

A followup to the CVQE algorithm called LCVQE [13] avoids checking all possible assignments for cannot-link constraints and its penalty calculations take into account coordinates of the involved instances in the violated constraint. However, this algorithm, similar to CVQE, works better in the presence of more must-link constraints [13] and has difficulty in satisfying cannot-link constraints. This limitation is due to the fact that cannot-link feasibility is an NP-hard problem.

Another important approach in the literature for semi-supervised clustering is spectral constrained clustering. Wang and Davidson [14] considered a degree of belief on pairwise instance level constraints. In this case, instead of considering constraints as hard (they must hold), constraints have an attached degree of belief, a real value between 0 and 1, where 1 represents a hard constraint. In the spectral method proposed in their paper, as opposed to some other methods, the aim is not to satisfy as many constraints as possible, because of possible inconsistency in constraints. Therefore, a threshold is defined allowing a fraction of constraints to remain unsatisfied. A lower bound is calculated for this threshold to guarantee finding a feasible solution. Zhi et al [15] incorporate more complex constraints in clustering. They show that any constraint represented using conjunctive normal form (CNF) can be formulated as a linear inequality. A specific percentage of constraints are allowed to remain unsatisfied in this framework.

Duong et al [16] introduced a constraint programming framework for constrained clustering that can incorporate both instance-level and cluster level constraints. Minimizing the maximum cluster diameter and the vector quantization error are two objectives. The constraint programming framework is able to find a globally optimum solution of the first criterion for data sets of size up to 1500.

Babaki et al [2] used a column generation framework for exactly solving the constrained clustering problem. The objective function used in this framework is similar to the k-means algorithm. However, this framework is slow, even for small data sets. To fasten, Babaki et al, initialize their column generation with the best solution found from running the COP algorithm 500 times. This can make the framework feasible for small datasets with a very large amount of constraints.

Our LCC algorithm, described in the next section, uses a Lagrangian relaxation strategy of increasing penalties for constraints which remain unsatisfied in subsequent clustering iterations. Such a scheme is loosely connected with the machine learning classification algorithm AdaBoost [17], which employs dynamic weights in a classifier ensemble. Some main differences are that our context is unsupervised, rather than supervised and that penalties (weights) are associated with constraints rather than classifiers.

3 Lagrangian Constrained Clustering Algorithm

In this section, we explain our Lagrangian constrained clustering algorithm (LCC) which can tackle the difficult NP-hard problem of constrained clustering incorporating both must-link and cannot-link constraints.

We first create super instances using transitive closure of must-link constraints. Hence we guarantee 100% satisfaction of must-link constraints. Then we tackle the difficult problem of satisfying cannot-link constraints using a Lagrange multipliers method.

We consider clustering of n instances $x_i, 1 \leq i \leq n$ into k clusters, with must-link constraints $(i, i') \in ML$ requiring that instance i and i' are in the same cluster, and cannot-link constraints $(i, i') \in CL$ which require that instance i and i' are not in the same cluster. Note that we treat ML and CL (and NL introduced later) as sets of *unordered* pairs. Let μ_j denotes the center of cluster j, i.e. $\mu_j = (\sum_{i=1..n, c_i=j} x_i)/|\{i \mid i \in 1..n, c_i = j\}|$, and let c_i be the cluster that instance i is assigned to. We assume a Euclidean distance function $D(x, x')$. Equation 3.1 represents a constrained clustering problem with instance-level constraints and k-means like objective function which aims to minimize the total distance of instances to their associated cluster center.

$$(3.1) \qquad min \quad Z = \sum_{j=1}^{k} \sum_{1 \leq i \leq n, c_i = j} D(x_i, \mu_j)^2$$
$$\text{subject to}$$
$$c_i = c_{i'} \qquad \forall (i, i') \in ML$$
$$c_i \neq c_{i'} \qquad \forall (i, i') \in CL$$

We first create super instances to handle the must-link constraints. Let $E = \{i = i' | (i, i') \in ML\}$ be equations representing the must link constraints. Let p map each instance i to a super instance j such that $(i, i') \in ML \rightarrow p(i) = p(i')$, any two must-link constrained instances are in the same super instance; and $p(i) = p(i') \rightarrow E \models i = i'$, any two instances in the same super instances are equal by transitivity from the must-link constraints. Let $w_j = |\{i | 1 \leq i \leq n, p(i) = j\}|$ be the weight of the super instance, i.e. number of original instances it represents, and $X_j = \sum_{1 \leq i \leq n, p(i)=j} x_i / w_j$ be the position of the super instance, the average of the instances it represent. We convert the cannot-link constraints to refer to the super instances, obtaining $NL = \{(p(i), p(i')) \mid (i, i') \in CL\}$.

We transform the constrained optimization problem of (3.1) to the unconstrained problem of equation (3.2) by using super instances (X) to enforce must-link constraints and adding a penalty term with a Lagrangian coefficient $\lambda_{(i,i')}$ for violated cannot-link constraint $(i, i') \in NL$.

$$(3.2) \qquad min \quad Z = \sum_{j=1}^{K} \sum_{c_i = j} w_i D(X_i, \mu_j)^2 + \sum_{(i,i') \in NL} \lambda_{(i,i')} viol(i, i', c)$$

The Lagrangian multiplier $\lambda_{(i,i')}$ associated with cannot-link constraint (i, i') is multiplied by the "degree of violation", $viol(i, i', c)$, of the cannot-link constraint (i, i') for the current assignment c to penalize the constraint violation.

Lets examine how we should measure the degree of violation of a cannot-link constraint. Given a clustering c then $swap(i, j, c)$ is the additional cost that will be added to the k-means objective if instance i is moved to cluster j from its current cluster c_i.

$$swap(i, j, c) = w_i D(X_i, \mu_j)^2 - w_i D(X_i, \mu_{c_i})^2$$

We define the violation of a cannot-link constraint to be 0 if the two instances are in seperate clusters, otherwise its the minimum of the swap costs for moving one of the two instances involved.

$$viol(i, i', c) = \begin{cases} 0 & c_i \neq c_{i'} \\ \min_{i'' \in \{i, i'\}, j \neq c_{i''}} swap(i'', j, c) & c_i = c_{i'} \end{cases}$$

Given a clustering c we can associate with a cluster j the set of instances i which are part of violated cannot-link constraints in NL whose minimum violation is determined by moving i to j from c_i. Let $conf(j, c) = \{\langle i, i' \rangle \mid (i, i') \in NL, viol(i, i', c) = swap(i, j, c)\}$ capture this set of instances and their associated violated constraint.[1] We can rewrite problem (3.2) as

$$\text{(3.3)} \qquad min \quad Z = \sum_{j=1}^{K} \sum_{c_i=j} w_i D(X_i, \mu_j)^2 + \sum_{j=1}^{K} \sum_{\langle i,i' \rangle \in conf(j,c)} \lambda_{(i,i')} swap(i,j,c)$$

by simply collecting violation terms associated with each cluster j.

In updating centers, we consider the objective (3.3) and conflict sets. To find the best cluster centres (and consequently the best assignment) to minimize the objective (3.3), we set the derivative of the objective Z over μ_j for each cluster j to zero and solve the equation 3.4. The solution to this equation (denoted by μ^*) is the center updating rule in the LCC algorithm.

(3.4)

$$\frac{\partial Z}{\partial \mu_j} = 0$$

$$\sum_{c_i=j} 2w_i(\mu_j - X_i) + \sum_{\langle i,i' \rangle \in conf(j,c)} 2w_i \lambda_{(i,i')} (\mu_j - X_i) = 0$$

$$\mu_j \sum_{c_i=j} w_i + \mu_j \sum_{\langle i,i' \rangle \in conf(j,c)} w_i \lambda_{(i,i')} =$$

$$\sum_{c_i=j} w_i X_i + \sum_{\langle i,i' \rangle \in conf(j,c)} w_i \lambda_{(i,i')} X_i$$

$$\mu_j^* = \frac{\sum_{c_i=j} w_i X_i + \sum_{\langle i,i' \rangle) \in conf(j,c)} w_i \lambda_{(i,i')} X_i}{\sum_{c_i=j} w_i + \sum_{\langle i,i' \rangle \in conf(j,c)} w_i \lambda_{(i,i')}}$$

Having established the formalism for the LCC algorithm, we provide some further intuition based on the pseudo code of LCC algorithm (Figure 1). The algorithm first creates super instances by merging instances based on transitive closure and updates the coordinates. Then, starting with a random assignment and initial centres (line 1-3), it start the main iterations till stopping criteria is met (lines 7-40). At each iteration, the priority assignment for each instance i is to its nearest centre c_i (line 11). In the case that the prior assignment causes any violation in cannot-link constraints, the algorithm decides about possible reassignments (lines 12-35). Instead of all possible reassignments of pairs $(i, i') \in NL | c_i = c_{i'}$, LCC just considers the second nearest centre to i and i' and compares the *swap* cost of applying such reassignments (lines 14-21) and defines the violation penalty based on the minimum swap cost (associated with index i''). Then it compares three possible assignments and chooses whether to accept the penalty and keep the constraint violated or make it satisfied (lines 22-35). However, for handling violated constraints in future iterations, we increase the penalty by increasing Lagrangian multipliers associated to the violated constraints if they remain violated ($\lambda_{(i,i')} = \alpha \lambda_{(i,i')}, \alpha > 1$). This increased penalty for difficult constraints helps in two ways: first, in updating centres, it causes the second closest centre for violated $(i, i') \in NL$ to move closer to one of the instances; second it helps reject the option of letting the constraint remain violated by increasing the violation cost. At the end of each iteration (line 39), it updates the centres based on the coordinates of the cluster members and the instances in the conflict set of each cluster. Updating centres is done according to equation (3.4). The algorithm keeps track of the best solution at each step (lines 37-39). As the main goal of LCC framework is to satisfy constraints, we consider this priority by defining best solutions based on the lexicographic order of the number of violated constraints and the k-means objective (sum of distances to cluster centres) We consider the stopping criteria as a fixed number of iteration where no improvement occurs and we also have a maximum number of iterations to help us for cases with slow convergence or inconsistent/infeasible constraints.

As our algorithm handles the must-link satisfaction problem in its first step, its main focus and advantage over existing algorithms is its ability to tackle the NP-hard problem of constrained clustering with cannot-link constraints. As opposed to the CVQE and LCVQE algorithms which are designed for the case of many must-link and a few cannot-link constraints [13], our method handles large numbers of must-link and cannot-link constraints easily and its performance increases with existence of more constraints of both kinds.

4 Experiments results

In this section, we present experimental results to evaluate the Lagrangian constrained clustering (LCC) method and compare it to some state of the art existing clustering algorithms which incorporate instance level constraints. Table 1 represents some information about the 12 real data sets from UCI Repository which we used in our experiments. In our experiments, we used a termination criteria of 25 iterations without improvement in the best solution and we also allowed a maximum of 100 iterations for the LCC algorithm. The penalty growth coefficient α was set to 2 for all data sets and experiments. All the experiments were performed on a Mac 2.7 GHz Intel Core i5 with 8GB RAM.

[1]We use angle brackets $\langle i, i' \rangle$ to emphasise this is an ordered pair.

```
LCC($X$, $NL$)
1.  Let $c$ be a random assignment of (super)instances to clusters
2.  for ($j \in 1..k$)
3.      $\mu_j = \sum_{1 \leq i \leq n, c_i = j} X_i / \sum_{1 \leq i \leq n, c_i = j} w_i$
4.  $bestV = |NL|$
5.  $bestO = \sum_{1 \leq i \leq n} w_i D(X_i, \mu_{c_i})$
6.  $bestc = c$
7.  while stopping criterion does not hold
8.      $V = 0$; %% number of violations
9.      $conf(j, c) = \{\}$
10.     for ($i \in 1..n$)
11.         $c_i = \text{indexmin}[D(X_i, \mu_j) \mid j \in 1..k]$
12.         Assign $i$ to $c_i$
12.     for ( $(i, i') \in NL$ where $c_i = c_{i'}$ )
13.         $h = c_i$
14.         $c'_i = \text{indexmin}[D(X_i, \mu_j)^2 \mid j \in 1..k, j \neq h]$
15.         $c'_{i'} = \text{indexmin}[D(X_{i'}, \mu_j)^2 \mid j \in 1..k, j \neq h]$
16.         $swap_i = swap(i, c'_i, c)$
17.         $swap_{i'} = swap(i', c'_{i'}, c)$
18.         if ($swap_i < swap_{i'}$)
19.             $i'' = i$; $i''' = i'$
20.         else
21.             $i'' = i'$; $i''' = i$
22.         %% Let the constraint remain violated
23.         $a = w_i \times D(X_i, \mu_h)^2 + w_{i'} \times D(X_{i'}, \mu_h)^2 + \lambda_{(i,i')} \times swap_{i''}$
24.         %% Force $i$ element to go to $c'_i$
25.         $b = w_i \times D(X_i, \mu_{c'_i})^2) + w_{i'} \times D(X_{i'}, \mu_h)^2$
26.         %% Force $i'$ element to go to $c'_{i'}$
27.         $c = w_i \times D(X_i, \mu_h)^2 + w_{i'} \times D(X_{i'}, \mu_{c'_{i'}})^2$
28.         if ($a \leq b \wedge a \leq c$)
29.             $\lambda_{(i,i')} = \alpha \times \lambda_{(i,i')}$
30.             $conf(c'_{i''}, c) = conf(c'_{i''}, c) \cup \{\langle i'', i''' \rangle\}$
31.             $V++$
32.         elseif ($b \leq a \wedge b \leq c$)
33.             $c_i = c'_i$
34.         else
35.             $c_{i'} = c'_{i'}$
36.     $O = \sum_{1 \leq i \leq n} w_i D(X_i, \mu_{c_i})^2$
37.     if ( $(V, O) < (bestV, bestO)$ )
38.         $bestV = V; bestO = O; bestc = c$
39.     update centres using Equation (3.4)
40. return $bestc$
```

Figure 1: Psuedo-code for the LCC algorithm for constrained clustering, starting from the super instances X and cannot-link constraints NL.

Table 1: Description of data sets

	# Instances	# Features	# clusters
Iris	150	4	3
Wine	178	13	3
Ionosphere	351	33	2
WDBC	569	30	2
KDD	600	60	6
Letter	600	16	3
Vehicle	846	18	4
Glass	214	9	6
Yeast	1484	8	10
Breast Tissue	106	9	6
Ecoli	336	7	8
Half Rings	400	2	2

4.1 Comparison to inexact methods We compare LCC with CVQE [6] and the spectral algorithm of Wang et al (CSP) [14]. For each benchmark of size n we generated $n/4$, $n/2$ and n constraints, equally divided into must-link and cannot-link constraints. For generating each of must-link (cannot-link) constraints, we pick a random instance and then randomly pick another instance from the same (different) cluster based on the ground truth (class labels). For each size of the constraint sets we created 5 different sets of constraints, and for each of these sets we ran each algorithm 8 times. Hence, the results shown in Tables 2 and 3 are the average and standard deviation of the results over these 40 independent executions. Run time of algorithms was different according to size of data sets and different constraint sets. The CSP spectral method was faster (maximum 1.5 minutes) than LCC (maximum 5 minutes) and CVQE (maximum 8 minutes except for a few cases). A timeout of 15 minutes was used for each algorithm. However, we were unable to get any result for some constraint sets on the Yeast data set for CVQE due to erroneous behaviour (— entries in tables).

In Table 2 we compare the methods in terms of their ability to satisfy constraints. We show for each of the three methods the average and standard deviation of number of must-link (#ML) and cannot-link (#CL) constraints violated in the partitions returned by the algorithm. The table clearly shows the LCC is superior in terms of satisfying constraints.

Since we have the ground truth of the data sets, for evaluating quality of partitioning, we use the Normalized Mutual Information of equation (4.5) which was proposed by Danon *et al* [18].

$$I_{norm}(A,B) = \frac{-2\sum_{i=1}^{CA}\sum_{j=1}^{CB} N_{ij}\log(N_{ij}N/N_{i.}N_{.j})}{\sum_{i=1}^{CA} N_{i.}\log(N_{i.}/N) + \sum_{j=1}^{CB} N_{.j}\log(N_{.j}/N)} \quad (4.5)$$

In Table 3 we compare the methods in terms of the normalized mutual information of the results they achieved compared to the ground truth. Since LCC directly enforces the must-link constraints, we might imagine that NMI results are inflated by the fact that directly enforce these ground truths. Similarly since we make use of the ground truth in the cannot-link constraints this may also inflate the NMI value. To counter this argument we also show the NMI value without considering instances involved in must-link constraint -ML and without considering instances involved in either must-link or cannot-link constraints -ML-CL. Hence for -ML-CL we consider the NMI only for instances with no background knowledge given by the constraints.

The results of Table 3 illustrate that the LCC method strictly improves in NMI as the number of constraints grow. In most cases, -ML-CL and -ML are also strictly improved by adding more constraints. Compared to the other methods LCC is almost always superior in NMI (exceptions are Wine and Breast Tissue) once we use n constraints.

4.2 Comparison to Exact method In this experiment we compare LCC with the state of the art Column Generation Constrained Clustering (CCCG) framework [2]. The CCCG framework optimizes the k-means objective satisfying all the constraints, and can prove optimality, which LCC cannot do. As the CCCG method is very slow without initialization, we initialized the algorithm with the best solution found by running the CVQE algorithm [6] 10 times. However, the framework is still very slow for small or moderate number of constraints. Here we used a timeout of 30 minutes. Table 4 presents the results of our experiment in terms of NMI on the Iris data set with different number of constraints. In cases marked — CCCG failed to find any feasible solution within 30 minutes. Otherwise for all cases both methods satisfied all constraints. We can see that, for large numbers of constraints, LCC could reach the optimal solution. For small and moderate numbers of constraints where CCCG struggles to find a solution, LCC obtains better quality solutions in much less time (around 1 minute). For larger number of constraints CCCG is superior, when the size of the problem is not too large.

4.3 Sensitivity Analysis on number of constraints In this experiments we evaluate sensitivity of constrained clustering methods' performance to the amount of background knowledge. Different number of constraints (equally contain ML and CL constraints) are randomly generated from ground truth based on the procedure explained previously. The performance

Table 2: Comparing number of violations

		CVQE		Spectral (CSP)		LCC	
	# constraints	# ML	# CL	# ML	# CL	# ML	# CL
Iris	[n/4]=37	1.1±0.8	0.6±0.9	0.4±0.5	12.9±1.7	**0.0**±0.0	**0.0**±0.0
	[n/2]=75	2.6±2.0	0.4±0.5	1.4±0.9	27.0±3.0	**0.0**±0.0	**0.0**±0.0
	[n]=150	7.5±2.5	2.1±1.2	15±6.1	42.6±4.6	**0.0**±0.0	**0.0**±0.0
Wine	[n/4]=44	0.9±0.8	0.2±0.4	1.3±1.1	17.6±1.9	**0.0**±0.0	**0.1**±0.2
	[n/2]=89	1.9±1.3	**0.3**±0.5	2.4±2.7	33.7±3.8	**0.0**±0.0	0.4±0.5
	[n]=178	2.7±2.3	**0.9**±0.9	11.3±6.2	66.4±15.0	**0.0**±0.0	1.3±1.2
Ionosphere	[n/4]=87	9.7±1.1	**1.0**±1.1	4.2±4.6	32.5±9.8	**0.0**±0.0	1.3±0.6
	[n/2]=175	20.7±4.7	6.2±1.6	28.6±5.1	46.1±14.3	**0.0**±0.0	**5.9**±3.6
	[n]=351	36.5±4.4	17.4±3.8	35±21.5	113.8±54.2	**0.0**±0.0	**7.1**±5.1
WDBC	[n/4]=142	4.6±0.7	2.0±1.1	9.6±13.6	53.2±20.4	**0.0**±0.0	**0.9**±0.6
	[n/2]=284	11.8±2.1	4.5±1.5	17.0±10.5	47.800	**0.0**±0.0	**0.4**±0.5
	[n]=569	20.4±4.5	16.6±4.0	31.6±25.8	138.6±123.2	**0.0**±0.0	**1.0**±1.2
KDD	[n/4]=150	7.4±2.2	**0.3**±0.4	4.1±1.9	65.5±2.9	**0.0**±0.0	**0.3**±0.6
	[n/2]=300	14.4±3.7	0.9±1.3	10.29±2.5	132.1±3.9	**0.0**±0.0	**0.3**±0.8
	[n]=600	31.2±5.4	3.3±1.8	44.2±9.6	223.8±29.4	**0.0**±0.0	**3.2**±3.7
Letter	[n/4]=150	8.3±2.3	**0.0**±0.2	0.8±1.0	70.3±0.8	**0.0**±0.0	**0.0**±0.2
	[n/2]=300	15.7±5.5	2.5±1.5	6.4±2.7	133.8±3.7	**0.0**±0.0	**0.6**±1.0
	[n]=600	24.7±5.5	3.4±2.0	30.4±12.4	261.6±12	**0.0**±0.0	**1.4**±1.6
Vehicle	[n/4]=211	51.7±4.7	**1.7**±0.8	2.4±1.8	98.0±1.9	**0.0**±0.0	**1.7**±1.5
	[n/2]=423	96.1±7.6	4.5±1.4	9.6±5.3	190.2±13.1	**0.0**±0.0	**4.1**±2.1
	[n]=846	185.6±8.7	**15.3**±2.0	31.4±23.8	378.6±31.9	**0.0**±0.0	30.1±6.5
Glass	[n/4]=53	8.3±1.9	0.6±0.6	4.6±3.9	18.2±2.5	**0.0**±0.0	**0.0**±0.2
	[n/2]=107	18.2±4.0	0.7±0.7	6.8±3.8	37.0±7.5	**0.0**±0.0	**0.2**±0.5
	[n]=214	33.9±5.2	3.0±1.7	15.6±7.9	68.3±13.0	**0.0**±0.0	**1.3**±1.7
Yeast	[n/4]=371	62.2±8.2	3.2±1.8	40.0±9.4	111.2±16.2	**0.0**±0.0	**0.8**±1.0
	[n/2]=742	—	—	91.6±25.4	193.2±50.5	**0.0**±0.0	**5.8**±4.4
	[n]=1484	—	—	252.8±24.1	293.2±21.3	**0.0**±0.0	**15.6**±2.8
Breast Tissue	[n/4]=26	4.5±1.7	**0.1**±0.3	1.4±1.5	5.0±2.8	**0.0**±0.0	**0.1**±0.3
	[n/2]=53	7.3±2.2	**0.3**±0.3	3.6±2.2	12.6±4.4	**0.0**±0.0	**0.3**±0.7
	[n]=106	13.4±2.7	**1.0**±0.8	14.1±3.0	16.9±6.9	**0.0**±0.0	1.4±1.6
Ecoli	[n/4]=84	5.4±2.2	**0.5**±0.5	5.2±3.4	12.2±1.6	**0.0**±0.0	1.0±0.4
	[n/2]=168	9.9±3.1	**1.8**±0.7	16.2±4.8	22.8±4.4	**0.0**±0.0	2.2±1.9
	[n]=336	22.6±6.8	**4.1**±2.0	41.0±6.3	25.1±9.5	**0.0**±0.0	5.0±3.1
Halfrings	[n/4]=100	1.6±1.3	**0.0**±0.0	3.4±2.1	4.0±2.1	**0.0**±0.0	0.4±0.2
	[n/2]=200	4.3±1.1	**0.1**±0.3	11.0±5.1	5.6±3.8	**0.0**±0.0	2.0±0.9
	[n]=400	7.7±2.8	1.1±1.5	19.4±12.6	13.4±8.4	**0.0**±0.0	**0.6**±0.3

of LCC, CVQE and the CSP spectral algorithm are demonstrated in terms of normalized mutual information value of the results compare to the ground truth. For each number of constraints, 10 different constraint sets are generated and we run each algorithm 5 times on each constraint set. Finally, the average and standard deviation of the results are represented in Figures 2 and 3 for Ionosphere and WDBC data sets respectively. Note that although column generation framework is designed to take the most advantage of background knowledge, it was too slow (due to data size and number of constraints) to be practical to use in this sensitivity analysis experiment. The Figures 2 and 3 show that although CVQE can compete with LCC in small number of constraints, LCC is superior to take advantage of background knowledge when number of constraints grows.

4.4 Sensitivity Analysis on incorrect constraints The LCC algorithm is designed to greedily satisfy all the constraints with the assumption that constraints are correct and should be satisfied. In this experiment we deliberately add a number of incorrect constraints to the constraint set obtained from the ground truth of Iris data set. Note that we add this experiment for completeness since false constraints violate the main assumption of LCC framework.

To generate test sets, we consider a constraint set with 100 constraints (equally divided to ML and CL constraints) and changed 1 to 10 randomly selected constraints to be disagree with the ground truth (1 to 10 percent) and generated 10 different noisy sets for each size. Figure 4 shows the average and standard deviations of normalized mutual information gained by each method. Clearly all methods are worsened by noise in the constraints, but LCC is the most affected,

Table 3: Comparing Normalized Mutual Information (NMI)

		CVQE			Spectral (CSP)			LCC		
	# const	NMI	-ML	-ML-CL	NMI	-ML	-ML-CL	NMI	-ML	-ML-CL
Iris	[n/4]=37	0.75±0.02	0.74±0.03	0.74±0.03	0.07±0.02	0.03±0.01	0.00±0.00	**0.79**±0.02	**0.77**±0.02	**0.76**±0.01
	[n/2]=75	0.77±0.02	0.74±0.03	0.74±0.05	0.09±0.03	0.01±0.01	0.00±0.00	**0.84**±0.03	**0.78**±0.04	**0.77**±0.04
	[n]=150	0.76±0.03	0.78±0.05	**0.82**±0.12	0.14±0.07	0.07±0.05	0.00±0.00	**0.88**±0.05	**0.79**±0.08	0.77±0.1
Wine	[n/4]=44	**0.82**±0.02	**0.82**±0.02	**0.83**±0.04	0.04±0.02	0.01±0.01	0.00±0.00	0.46±0.03	0.46±0.03	0.44±0.03
	[n/2]=89	**0.83**±0.04	**0.82**±0.06	**0.82**±0.06	0.06±0.03	0.02±0.01	0.00±0.00	0.50±0.05	0.44±0.07	0.44±0.07
	[n]=178	**0.86**±06	**0.82**±0.08	**0.78**±0.08	0.07±0.05	0.02±0.02	0.00±0.00	0.67±0.03	0.50±0.08	0.47±0.13
Iono	[n/4]=87	**0.17**±0.03	**0.15**±0.02	**0.11**±0.01	0.06±0.07	0.06±0.6	0.05±0.05	0.12±0.02	0.10±0.02	0.10±0.02
	[n/2]=175	0.19±0.02	**0.16**±0.03	**0.10**±0.03	0.12±0.07	0.08±0.05	0.07±0.04	**0.20**±0.10	0.13±0.06	0.08±0.04
	[n]=351	0.25±0.03	0.19±0.04	0.17±0.10	0.18±0.23	0.17±0.24	0.14±0.16	**0.62**±0.10	**0.36**±0.13	**0.20**±0.13
WDBC	[n/4]=142	0.60±0.03	**0.58**±0.03	**0.59**±0.03	0.13±0.23	0.13±0.23	0.13±0.23	**0.62**±0.12	0.55±0.11	0.47±0.11
	[n/2]=284	0.64±0.01	0.60±0.03	**0.64**±0.02	0.43±0.25	0.40±0.24	0.40±0.24	**0.77**±0.03	**0.68**±0.03	0.57±0.04
	[n]=569	0.67±0.05	0.62±0.09	**0.69**±0.08	0.33±0.29	0.33±0.29	0.26±0.24	**0.90**±0.03	**0.79**±0.05	0.56±0.05
KDD	[n/4]=150	**0.77**±0.02	**0.77**±0.02	**0.77**±0.02	0.03±0.01	0.01±0.01	0.00±0.00	0.68±0.04	0.67±0.04	0.69±0.04
	[n/2]=300	**0.77**±0.02	**0.77**±0.02	**0.78**±0.02	0.03±0.01	0.02±0.01	0.00±0.00	0.72±0.04	0.70±0.04	0.72±0.04
	[n]=600	0.79±0.03	**0.77**±0.03	**0.80**±0.03	0.10±0.05	0.04±0.02	0.00±0.01	**0.81**±0.01	0.74±0.05	0.76±0.03
Letter	[n/4]=150	**0.62**±0.06	**0.61**±0.06	**0.61**±0.05	0.02±0.01	0.01±0.00	0.00±0.00	0.58±0.10	0.56±0.10	0.56±0.09
	[n/2]=300	0.65±0.04	0.62±0.06	0.61±0.06	0.03±0.01	0.01±0.01	0.00±0.00	**0.72**±0.10	**0.64**±0.10	**0.63**±0.11
	[n]=600	0.73±0.04	0.67±0.05	0.66±0.07	0.04±0.01	0.01±0.01	0.00±0.00	**0.86**±0.05	**0.73**±0.09	**0.71**±0.11
Vehicle	[n/4]=211	0.10±0.01	0.11±0.01	0.10±0.01	0.01±0.00	0.01±0.00	0.00±0.00	**0.17**±0.02	**0.16**±0.03	**0.17**±0.03
	[n/2]=423	0.11±0.02	0.11±0.02	0.11±0.03	0.02±0.01	0.01±0.00	0.00±0.00	**0.18**±0.02	**0.15**±0.03	**0.15**±0.04
	[n]=846	0.11±0.00	0.10±0.01	0.12±0.03	0.02±0.01	0.01±0.01	0.00±0.00	**0.25**±0.03	**0.14**±0.02	**0.15**±0.04
Glass	[n/4]=53	0.36±0.02	0.35±0.03	0.35±0.00	0.12±0.03	0.07±0.02	0.04±0.03	**0.40**±0.04	**0.37**±0.03	**0.37**±0.03
	[n/2]=107	0.37±0.03	0.33±0.03	0.34±0.00	0.14±0.07	0.06±0.03	0.03±0.05	**0.43**±0.04	**0.36**±0.03	**0.36**±0.03
	[n]=214	0.37±0.04	0.33±0.04	0.28±0.00	0.18±0.07	0.04±0.04	0.00±0.00	**0.51**±0.04	**0.36**±0.04	**0.32**±0.04
Yeast	[n/4]=371	0.28±0.01	**0.24**±0.01	**0.21**±0.01	0.09±0.01	0.04±0.02	0.03±0.02	**0.29**±0.02	0.22±0.02	**0.21**±0.02
	[n/2]=742	—	—	—	0.11±0.04	0.04±0.01	0.03±0.01	**0.34**±0.02	**0.22**±0.02	**0.19**±0.02
	[n]=1484	—	—	—	0.14±0.01	0.04±0.01	0.03±0.00	**0.41**±0.02	**0.21**±0.02	**0.19**±0.02
Breast T	[n/4]=26	**0.54**±0.02	**0.55**±0.02	**0.57**±0.00	0.19±0.08	0.11±0.11	0.07±0.16	0.37±0.03	0.36±0.04	0.38±0.05
	[n/2]=53	**0.54**±0.02	**0.57**±0.05	**0.59**±0.10	0.20±0.08	0.14±0.11	0.06±0.17	0.41±0.03	0.39±0.03	0.41±0.04
	[n]=106	**0.59**±0.02	**0.59**±0.03	**0.60**±0.10	0.27±0.12	0.17±0.17	0.18±0.29	0.53±0.04	0.43±0.04	0.56±0.07
Ecoli	[n/4]=84	**0.60**±0.03	**0.60**±0.03	**0.59**±0.00	0.19±0.04	0.10±0.03	0.09±0.03	0.58±0.03	0.57±0.02	0.56±0.02
	[n/2]=168	0.61±0.03	**0.59**±0.03	**0.60**±0.00	0.18±0.02	0.11±0.02	0.11±0.01	**0.62**±0.04	0.58±0.02	**0.60**±0.02
	[n]=336	0.62±0.03	**0.58**±0.02	**0.59**±0.00	0.25±0.03	0.17±0.03	0.21±0.03	**0.67**±0.03	0.54±0.03	0.58±0.02
Halfrings	[n/4]=100	**0.57**±0.02	**0.48**±0.02	**0.39**±0.04	0.38±0.21	0.32±0.18	0.25±0.19	0.39±0.02	0.29±0.11	0.23±0.07
	[n/2]=200	**0.62**±0.03	**0.50**±0.02	**0.29**±0.03	0.47±0.21	0.38±0.25	0.23±0.29	0.50±0.06	0.33±0.04	0.18±0.05
	[n]=400	0.73±0.03	**0.46**±0.07	**0.15**±0.06	0.58±0.13	0.36±0.15	0.21±0.12	**0.75**±0.04	0.37±0.04	0.10±0.05

Table 4: Comparing LCC and CCCG algorithms: entries marked∗ are proved optimal, no solution was found at timeout for − entries

	CCCG		LCC	
#C	NMI	time (s)	NMI	time (s)
37	0.75	1800	0.80	1.2
75	–	1800	0.84	1.5
150	–	1800	0.88	1.6
200	0.92∗	137.5	0.92	2.2
250	0.92 ∗	3.2	0.92	1.9
300	1∗	0.5	1	2.0
400	1 ∗	0.2	1	2.4

since it concentrates on satisfying constraints. Note that we didn't include column generation framework in this experiment but it is the kind of method which would be mostly affected by noise constraints due to its assumption (correctness of all constraints) similar to LCC.

5 Conclusion

We have developed a k-means style algorithm for constrained clustering which is targeted at satisfying as many of the constraints as possible, as opposed to simply improving the k-means objective. We compared performance of the proposed LCC framework with some state of the art exact and inexact algorithms. It is very effective at satisfying constraints in comparison to approximate algorithms, and scales much better than existing methods that guarantee that all constraints are satisfied. Sensitivity analysis also demonstrates that LCC performs strongly in presence of more constraints. Incorporating cluster level constraints is an interesting direction of future work for the LCC algorithm.

References

[1] B. Wu, Y. Zhang, B.-G. Hu, and Q. Ji, "Constrained

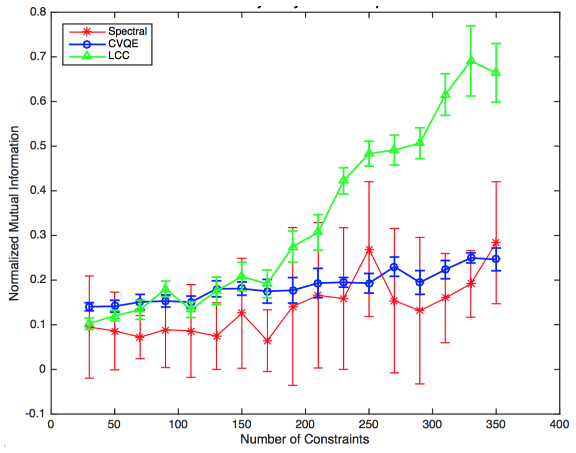

Figure 2: Sensitivity analysis on number of constraints (Ionosphere)

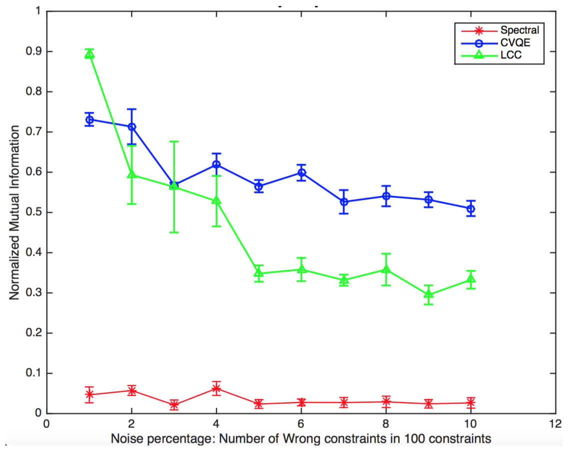

Figure 4: Sensitivity analysis on number of incorrect constraints (Iris)

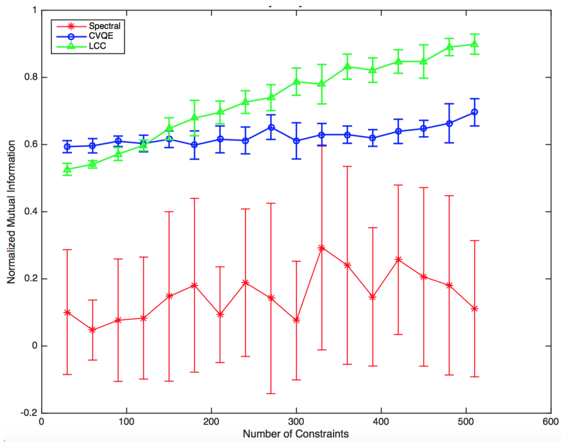

Figure 3: Sensitivity analysis on number of constraints (WDBC)

clustering and its application to face clustering in videos," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on.* IEEE, 2013, pp. 3507–3514.

[2] B. Babaki, T. Guns, and S. Nijssen, "Constrained clustering using column generation," in *Integration of AI and OR Techniques in Constraint Programming.* Springer, 2014, pp. 438–454.

[3] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl *et al.*, "Constrained k-means clustering with background knowledge," in *ICML*, vol. 1, 2001, pp. 577–584.

[4] Z. Lu, M. Carreira-Perpinan *et al.*, "Constrained spectral clustering through affinity propagation," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on.* IEEE, 2008, pp. 1–8.

[5] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering." in *SDM*, vol. 4. SIAM, 2004, pp. 333–344.

[6] I. Davidson and S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm." in *SDM*, vol. 5. SIAM, 2005, pp. 201–211.

[7] I. Davidson and S. Basu, "A survey of clustering with instance level constraints," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, pp. 1–41, 2007.

[8] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of np-completeness. 1979," *San Francisco, LA: Freeman*, 1979.

[9] M. Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, pp. 10–21, 1985.

[10] K. M. Choi, J. H. Lee, and P. J. Stuckey, "A lagrangian reconstruction of genet," *Artificial Intelligence*, vol. 123, no. 1, pp. 1–39, 2000.

[11] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods.* Academic press, 2014.

[12] S. Basu, A. Banerjee, and R. J. Mooney, "Active semi-supervision for pairwise constrained clustering." in *SDM*, vol. 4. SIAM, 2004, pp. 333–344.

[13] D. Pelleg and D. Baras, "K-means with large and noisy constraint sets," in *European Conference on Machine Learning.* Springer, 2007, pp. 674–682.

[14] X. Wang and I. Davidson, "Flexible constrained spectral clustering," in *16th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2010, pp. 563–572.

[15] W. Zhi, X. Wang, B. Qian, P. Butler, N. Ramakrishnan, and I. Davidson, "Clustering with complex constraints-algorithms and applications." in *AAAI*, 2013.

[16] K.-C. Duong, C. Vrain *et al.*, "A declarative framework for constrained clustering," in *Machine Learning and Knowledge Discovery in Databases.* Springer, 2013, pp. 419–434.

[17] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, p. 119139, 1997.

[18] A. D.-G. Leon Danon and A. Arenas, "The effect of size heterogeneity on community identification in complex networks," *Journal of Statistical Mechanics: Theory and Experiment*, p. P11010, 2006.