Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive 5 - 38123 Povo - Trento (Italy)
http://disi.unitn.it

A Survey of Semi-Supervised Clustering Algorithms: from a priori scheme to interactive scheme and open issues

List of authors:
Duy Tin Truong
Roberto Battiti

July 2013

Technical Report # DISI-13-003
Version 1.0

Published also: …

A Survey of Semi-Supervised Clustering Algorithms: from a priori scheme to interactive scheme and open issues

Duy Tin Truong - Roberto Battiti
University of Trento

July 12, 2013

Contents

Abstract

In the last 10 years, semi-supervised clustering (SSC) or clustering with side information has received significant attention from researchers because of its success in many applications like document, image clustering, etc. SSC has been shown to improve the clustering performance substantially with just few constraints or labelled data points as side information which are provided by an expert or an oracle system. Most works have been done so far can be classified into one of two SSC schemes: the a-prior scheme, and the interactive scheme. This survey will cover these two schemes together with the important algorithms in each scheme. Finally, the open issues will also be summarized in the survey.

1 Introduction

Semi-supervised clustering (SSC) is the problem of clustering unlabelled data with the support of the *side information* provided by a supervisor (who can be an expert or an oracle system). And because of its great success in recent years, SSC has received significant attention from researchers. The side information has been shown to guide the clustering algorithms towards the desired clustering solutions or help the clustering algorithms escape from the local minima effectively. The side information does contribute not only to the performance improvement but also to the complexity reduction. An example is the car land identifying problem from GPS data where the goal is to cluster data points into different lanes [51]. This is a difficult clustering problem for the well-known clustering algorithm *KMEANS* because the lane clusters have a very special shape which is very elongated and parallel to the road centerline. And the *KMEANS* with constraints has achieved the accuracy of 98.6% whereas the accuracy of the *KMEANS* with no constraints is only 58% [51]. Some other applications of SSC can be found in [16].

The works that have been done so far can be classified into one of the following two schemes: the a-prior scheme, the interactive scheme. In the a-priori scheme, the side information is given once before executing the SSC algorithm while in the interactive scheme, the side information is collected iteratively by interacting with the supervisor. Although two excellent surveys by Davidson et al. [16] and Basu et al. [7] have covered main aspects of the a-priori scheme, there is still no surveys that cover also the other scheme. Besides, some recent important algorithms are also missing from these surveys. This survey comes to fill in that need with the hope that it can present not only a more general view but also a deeper view of this field for new researchers. The algorithms presented in this paper are grouped into common techniques for easy comparison. The pseudo-code as well as the advantages and disadvantages of each algorithm will be presented clearly. In addition, the open issues will be also summarized in the survey.

The outline of this paper will be as follows. In Section 2, two schemes and the algorithms in each scheme will be briefly introduced. Then, Section 3 will present the algorithms of the a-priori scheme classified based on different types of constraints. Next, Section 4 discusses in detail the interactive SSC algorithms. Finally, Section 5 concludes this survey by summarising the open issues in this field.

2 Two Schemes

Currently, there are two schemes for SSC which are the a-priori scheme, and the interactive scheme. They are basically different by the way the side information is collected in each scheme. In the first scheme, all side information is given once before the SSC algorithm is executed while in the second scheme, side information is collected iteratively by interacting with the supervisor.

2.1 A Priori Scheme

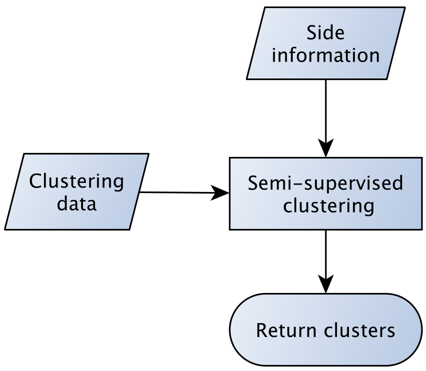


Figure 1: A Priori Scheme

In the a priori scheme (shown in Fig. 1), the SSC algorithm reads all side information once and uses these information to improve the clustering performance. Many works following this scheme have been done in literature and split into different types of side information like labelled data, instance-level or cluster-level constraints.

Several techniques that utilizes the side information in the form of labeled data are:

- *Seeded K-Means* uses labeled examples to initialize the cluster centers [4].
- *Constrained K-Means* also initializes the cluster centers by labelled data like *Seeded K-Means* but keeps the labels of examples in the side information unchanged in the assignment step of the clustering process [4].

The algorithms which uses instance-level constraints provided by users to improve the clustering performance are divided into three groups:

- Constraint-Based Clustering: in this group, the original clustering algorithms are modified to integrate the constraints, e.g. in a constrained agglomerative clustering algorithm, two clusters are only merged if the merging of two clusters does not violate constraints [16], or adding the penalty of violating the constraints into the objective function of *KMEANS* [14, 37, 5]. The clustering solutions must satisfy completely the constraints [51, 44] or some constraints can be violated [14, 37, 5]. Also, two dominant approaches of the algorithms in this group are extending the objective function of *KMEANS* for integrating constraints [14, 37, 5] or adding constraints into prior distributions of probabilistic clustering frameworks such that the clustering solutions which satisfy constraints are given higher scores to be selected [44, 33, 6].
- Distance-Based Clustering: in this group, only the distance metric is changed such that if two points are constrained to be in the same cluster, their distance should be smaller than the distance of two points constrained to be in different clusters [52, 28].
- Unified framework for constraint-based and distance-based clustering is also proposed by Basu et al. [6].

In addition, cluster-level constraint based algorithms discussed in this paper are divided into two main problems:

- Balanced-Clustering: where the constraint is that the variance of cluster sizes is as small as possible. Two algorithms for this problem are proposed by Banerjee et al. [1] and Demiriz et al. [18].
- Non-Redundant Clustering: in this problem, the constraint is given as a clustering result and the goal is to find the new clustering result

which is as different as possible from the given clustering result. This problem has been introduced and solved by Goldek et al. [24, 23].

The details of the algorithms mentioned in this section are presented in Section 3.

2.2 Interactive Scheme

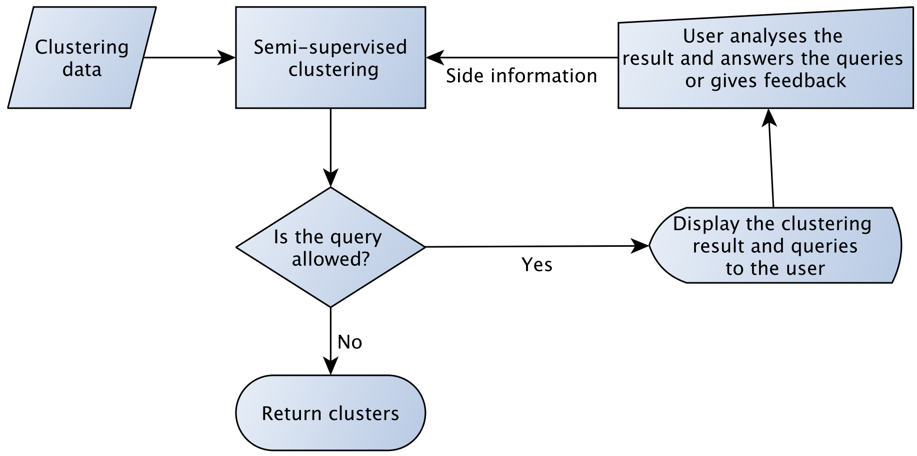


Figure 2: Interactive Scheme

In this interactive scheme (illustrated in Fig. 2), the SSC algorithm presents the clustering result and a query to a supervisor who can be a user or an oracle system. Then the supervisor studies the result and provides feedback to the SSC algorithm. The SSC algorithm in turn analyses the feedback and adapts this information to bias the clustering process. The interaction between the SSC algorithm and the supervisor is stopped when some convergence condition is satisfied. The feedback can be collected in two following ways based on the role of the supervisor and the SSC algorithm. If the supervisor plays the active role, then he/she actively provides the constraints to the SSC algorithm. In the case that the SSC algorithm is the active role, the SSC algorithm will pose queries to the supervisor, and the supervisor is supposed to answer these queries. The second approach has been shown to outperform the first approach in literature. The reason is that the first approach requires the supervisor must know which are the most informative constraints to supply for the SSC algorithm while in the second approach, this difficult task is on the side of the SSC algorithm, and it is better if the SSC algorithm is allowed to ask what it is not clear than passively receives irrelevant feedback from the supervisor. The algorithms in the first approach will be referred as the *passive* SSC algorithms, while the ones in the second approach will be called the *active* SSC algorithms.

So far, only few works have been done in this scheme. A interactive SSC algorithm integrates the constraints by changing the distance metric [12, 11] and other active SSC algorithms which uses the farthest distance [5], information gain [29], density [54] and co-association confidence [26] to select the most informative constraints will be discussed in Section 4.

3 A priori scheme

The algorithms of the a-priori scheme will be divided into two main groups based on two different types of side information which are labelled data or constraints.

3.1 Semi-Supervised Clustering Algorithms with Labels

Following the notation in [3], the problem of SSC with labeled data provided by users as side information is defined as follows. Given a dataset X, the goal is to split this dataset into K disjoint clusters $\{X_h\}_{h=1}^{K}$ such that some objective is minimized (often locally). Let $S \in X$ be the subset of data objects and called the *seed* set. The side information is given as follows: for each $x_i \in S$, the label $y_i = h$ of x_i denotes the cluster X_h which x_i belongs to. The seed set S is partitioned into L disjoint set $\{S_h\}_{h=1}^{L}$ where $L \leq K$. If $L = K$, the seed set is called complete. Otherwise, it is the case of incomplete seeding.

Basu et al. proposed two versions of KMeans that make use of labeled data as side information for improving the KMeans performance [4]. In the first algorithm *Seeded-KMeans*, the seed set is used to initialize cluster centers. Each cluster center μ_h is computed as the mean of data objects with the label of h in the seed set. If for some cluster X_h, there is no labelled data objects belonging to it, its center is initialized by random perturbations of the global center. And then the KMeans algorithm is applied on the whole dataset as usual. The idea of *Seeded-KMeans* is that a good seed set can guide KMeans towards a good region of search space. Algorithm 1 shows the pseudo code of *Seeded-KMeans*.

In the *Seeded-KMeans*, the cluster memberships of data objects in the seed set can be changed in the assignment step of KMeans. Therefore, in order to keep these memberships unchanged, the data objects in the seed set must be skipped in the assignment step. This modification leads to

Algorithm 1: Seeded-KMeans

Input : Dataset $X = \{x_i\}_{i=1}^{N}, x_i \in R^D$, seed set $S = \cup_{h=1}^{L} S_h$
Output: K disjoint clusters $\{X_h\}_{h=1}^{K}$ of X such that K-Means objective function is optimized

```
begin
    1. Initialize cluster centers:
    t = 0
    for h from 1 to K do
        if ∃x_i ∈ S : y_i = h then
            μ_h^(0) = 1/|S_h| Σ_{x∈S_h} x
        else
            μ_h^(0) = random perturbations of the global center
        end
    end
    2. K-Means:
    repeat
        2.1 assign_cluster: Assign each data object x_i to the nearest
        cluster h* (the set X_{h*}^(t+1)) where h* = argmin_{h∈{1,...,K}} ||x_i − μ_h^(t)||^2
        2.2 estimate_means: μ_h^(t+1) = 1/|X_h^(t+1)| Σ_{x_i∈X_h^(t+1)} x_i
        t = t + 1
    until convergence ;
end
```

Where:

$$\mu_h^{(0)} = \frac{1}{|S_h|} \sum_{x \in S_h} x$$

$$h^* = \underset{h \in \{1,\dots,K\}}{\operatorname{argmin}} ||x_i - \mu_h^{(t)}||^2$$

$$\mu_h^{(t+1)} = \frac{1}{|X_h^{(t+1)}|} \sum_{x_i \in X_h^{(t+1)}} x_i$$

the *Constrained-KMeans* illustrated in Algorithm 2. When the seed set is noise-free or the user does not want the change in the labels of the seed set, *Constrained-KMeans* is more suitable than *Seeded-KMeans*. However, if the seed set is noisy, *Seeded-KMeans* is supposed to be better because it does not need to keep the labels unchanged and then the noisy labels can be removed by KMeans.

3.2 Semi-Supervised Clustering Algorithms with Constraints

In many applications, the labeled data is not available whereas the constraints between instances or the constraints on clusters are easier to collect. Constraints can be divided into instance-level and cluster-level constraints.

3.2.1 Instance-Level Constraints

Instance-level constraints, also called pairwise constraints, are the constraints between data objects. There are two types of instance-level constraints which are *must-link* and *cannot-link* introduced by Wagstaff [51]. A *must-link* $c_=(x, y)$ or a *cannot-link* $c_{\neq}(x, y)$ constraint between two objects x and y means that these two objects must or must not be in the same cluster, respectively. The must-link constraint is an equivalence relation because it is reflexive, symmetric and transitive [16]. Besides, cannot-link constraints can be entailed from connected components CC_i where each connected component CC_i is a completely connected subgraph by must-link constraints. Two important properties of *must-link* and *cannot-link* constraints are stated formally as follows [16]:

Observation 1 ***Must-link constraints are transitive.*** *Let CC_i and CC_j be connected components (by must-link constraints), and let x and y be the instances in CC_i and CC_j respectively. Then $c_=(x, y), x \in CC_i, y \in CC_j \implies c_=(a, b), \forall a, b : a \in CC_i, b \in CC_j$.*

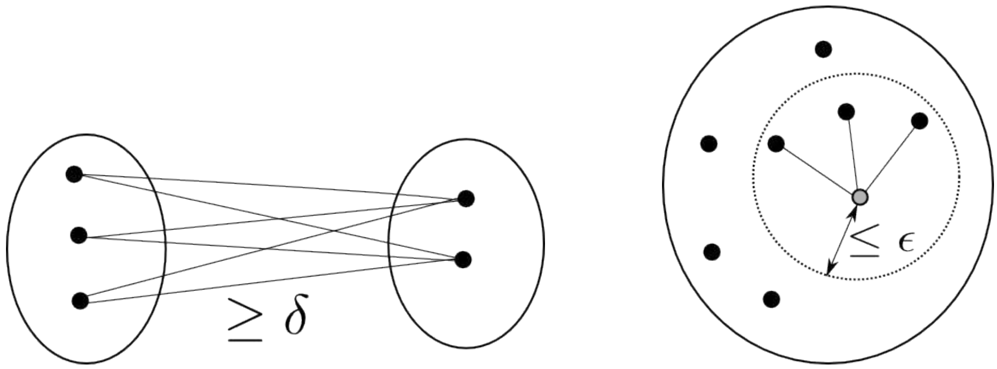
Observation 2 ***Cannot-link Constraints can be entailed.*** *Let CC_i and CC_j be connected components (by must-link constraints), and let x and y be the instances in CC_i and CC_j respectively. Then $c_{\neq}(x, y), x \in CC_i, y \in CC_j \implies c_{\neq}(a, b), \forall a, b : a \in CC_i, b \in CC_j$.*

Based on must-link and cannot-link constraints, Davidson and Ravi defines two other types of constraints: σ-constraint and ϵ-constraint [14] as illustrated in Fig. 3 (modified from [16]). σ-constraint (also called *minimum*

Algorithm 2: Constrained-KMeans

Input : Dataset $X = \{x_i\}_{i=1}^{N}, x_i \in R^D$, seed set $S = \cup_{h=1}^{L} S_h$
Output: K disjoint clusters $\{X_h\}_{h=1}^{K}$ of X such that K-Means objective function is optimized

begin
  1. Initialize cluster centers:
  $t = 0$
  for h *from* 1 *to* K **do**
    if $\exists x_i \in S : y_i = h$ **then**
      $\mu_h^{(0)} = \frac{1}{|S_h|} \sum_{x \in S_h} x$
    else
      $\mu_h^{(0)}$ = random perturbations of the global center
    end
  end
  2. Modified K-Means:
  repeat
    2.1 assign_cluster:
    for *each* $x_i \in X$ **do**
      if $x_i \in S$ **then**
        Assign x_i to the cluster h where $h = y_i$.
      else
        Assign data object x_i to the nearest cluster h^* (the set $X_{h^*}^{(t+1)}$) where $h^* = \underset{h \in \{1,\ldots,K\}}{\operatorname{argmin}} ||x_i - \mu_h^{(t)}||^2$
      end
    end
    2.2 estimate_means: $\mu_h^{(t+1)} = \frac{1}{|X_h^{(t+1)}|} \sum_{x_i \in X_h^{(t+1)}} x_i$
    $t = t + 1$
  until *convergence* ;
end



(a) δ-constraint (b) ϵ-constraint

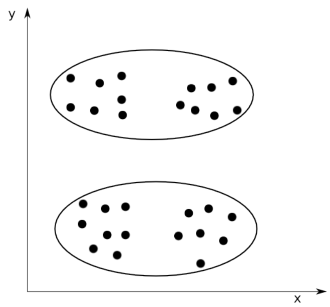
Figure 3: δ-constraint and ϵ-constraint

separation constraint) requires that any pair of points which are in two different clusters must have a distance greater than or equal to σ. σ-constraint is shown to be equivalent to a conjunction of must-link constraints between all instances with the distances less than σ. As for ϵ-constraint, for each point x in a cluster, there must exist a neighbour y of x, such that the distance between x and y is at most ϵ. In [14], ϵ-constraint is proven to be a disjunction of instance level must-link constraints.

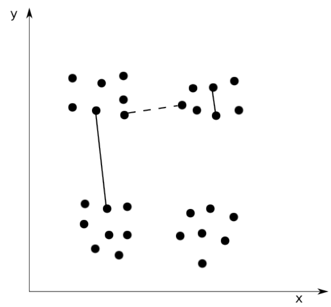
Typically, the instance-level constraints are exploited in the SSC algorithms in two ways. In the first approach, the constraints are used to guide the search strategy of the original clustering algorithm towards the clustering solutions in which these constraints are satisfied as many as possible. In the second approach, the distance function of the original clustering algorithm is adjusted according to the constraints in such a way that the points in the must-link pairs have small distances whereas the points in the cannot-link pairs are far from each other. Based on this classification, the existing SSC algorithms can be split into two classes which are *constrained-based* and *distance-based* clustering [16].

Constraint-Based Clustering

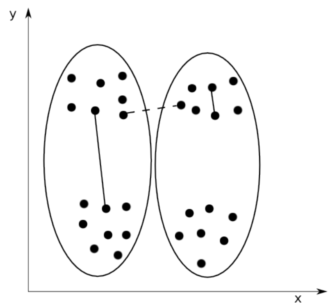
In this approach, the original clustering algorithm is modified to integrate the constraints so that the search strategy is biased towards the solutions which respect these constraints as many as possible. These constraints can be respected strictly or partially depending on the different clustering algorithms. Fig. 4 shows an example of a constraint-based clustering algorithm that satisfies all constraints. At the beginning without constraints, the clustering solution will be as in Fig. 4(a). In order to satisfy the constraints as in Fig. 4(b), the clustering solution will be as in Fig. 4(c). The SSC algo-



(a) Input instances without constraints



(b) Input instances and constraints. Must-link constraints are denoted as solid lines, and cannot-link constraints are denoted as dashed lines.



(c) A clustering solution that satisfies all constraints

Figure 4: Example of a constraint-based clustering algorithm

rithms in this section are divided into hierarchical clustering and partitioning clustering.

The first part of this section will present the agglomerative hierarchical clustering algorithms. Hierarchical Clustering (HC) is widely used in many areas of science to describe the hierarchical structure of data. The goal of HC is to construct a cluster hierarchical or a tree of clusters, also known as *dendrogram*, from data objects. HC algorithms are mainly categorized into: agglomerative (bottom-up) and divisive (top-down) approach. The agglomerative approach starts with singleton clusters (each singleton cluster is a data object) and recursively merge the two most similar clusters to larger clusters until the desired number of clusters is achieved. In contrast, the divisive approach starts with a cluster consisting all data objects, and successively splits each cluster into small clusters until a stopping condition is satisfied. Until now, only the agglomerative clustering is adapted to work with side information. The general framework of the agglomerative clustering algorithms is summarized in Algorithm 3.

Algorithm 3: Agglomerative Clustering

Start with singleton clusters.
repeat
  Calculate the similarity between clusters.
  Merge the two closest (most similar) clusters into a new cluster.
until *the desired number of clusters is achieved* ;

As can be seen from Algorithm 3, the agglomerative clustering algorithms are basically different from each other by their distance functions. There are two main strategies to compute the distance between two clusters: *graph methods* and *geometric methods*. In graph methods, the distance $D(C_i, C_j)$ between two clusters C_i, C_j is calculated by considering the minimal (*single linkage*), average (*average linkage*), or maximal (*complete linkage*) distance of all object pairs (x, y) where $(x \in C_i, y \in C_j)$. SLINK [45], Voorhees' algorithm [48], CLINK [17] are one of the first algorithms in data mining which implement single linkage, average linkage, and complete linkage, respectively. In geometric methods, each cluster is represented by its geometric center and the distance between clusters is calculated based on cluster centers.

The agglomerative clustering has been adapted to work with side information, called *Constrained Agglomerative clustering*, by Davidson et al. [16]. The framework of constrained agglomerative clustering is presented in Algorithm 4. The condition *mergeable clusters* of the While loop means the merging of these clusters does not violate the cannot-link constraints. Note that because of constraints, it is not always possible to achieve a desired number of clusters. An effort which tries to reduce the average complexity of computing distances between two clusters using a centroid distance function is proposed by Davidson et al. [16]. The idea is to make use of the triangle inequality. The triangle inequality of three instances a, b, c is: $|D(a, b) - D(b, c)| \leq D(a, c) \leq D(a, b) + D(c, b)$ where D is a metric function. Define the γ *constraint* as a restriction such that two clusters with the distance greater than γ will not be merged. Then the combination of the γ *constraint* with the triangle inequality can reduce the computational complexity in the following case. Given three cluster centroids a, b, c and the goal is to determine the two closest clusters to merge. If $D(a, b)$ and $D(a, c)$ are computed and $\gamma < |D(a, b) - D(a, c)|$, then it can be inferred that $\gamma < |D(a, b) - D(a, c)| \leq D(b, c)$. It means that the two clusters b and c cannot be merged according to the γ *constraint* because their distance exceeds γ. In other words, the lower bound of the distance between two cluster centroids b, c can be computed given the distances of these two cluster cen-

Algorithm 4: Constrained Agglomerative Clustering

Input : Dataset $X = \{x_i\}_{i=1}^{N}$, must-link constraints $C_=$, cannot-link constraints $C_{\neq}$.

Output: $Dendrogram_i, i = k_{min}, ..., k_{max}$ such that each level in the dendrogram satisfies all constraints.

begin

Construct the connected components from the must-link constraints in $C_=$: $M_1, M_2, ..., M_r$.

Let $X_1 = X - (\cup_{i=1}^{r} M_i)$. Let $k_{max} = r + |X_1|$.

Construct $Dendrogram_{k_{max}}$ including: r clusters $\pi_1 = M_1, ..., \pi_r = M_r$ and $|X_1|$ singleton clusters $\pi_{r+i}, i = 1, ..., |X_1|$, each for an instance in X_1.

Set $t = k_{max}$.

while *there exists a pair of mergeable clusters* **do**

Select a closest pair π_l and π_m according to the specific distance function.

Merge π_l into π_m, resulting in $Dendrorgram_{t-1}$.

$t = t - 1$.

end

$k_{min} = t$.

return $\{Dendrogram_i\}$

end

troids to another cluster centroid a. The function calculating the distance between any two cluster centroids using the γ *constraint* and the triangle inequality is shown in Algorithm 5. In this function, the distances between the pivot (chosen as the first centroid in this case) and all other centroids are computed. For any two cluster centroids (different from the pivot), if the difference between the distances of them to the pivot is greater than γ, then these two clusters cannot be joined and the computation of their distance is skipped. Their distance in this case is assigned as $\gamma + 1$. However, how to select the best pivot (instead of the first centroid) is still an open question. Besides, in the worst case when the difference between the distances of any two cluster centroids to the pivot is less than γ, there is no performance improvement obtained from the algorithm.

Another approach which uses the unsupervised proximity-based clustering and integrates the constraints by changing the distance matrix is proposed by Klein et al [31]. In this approach, the distance matrix between all data points is modified such that two points of a must-link constraint have a small distance whereas two points of a cannot-link constraint have a large distance.

Algorithm 5: Distance function using the γ constraint and the triangle inequality.

```
Input  : γ, cluster centroids {c_i}_{i=1}^K
Output: D_{c_i,c_j}, ∀i, j.
begin
    for i = 2 to n − 1 do
        D_{c_1,c_i} = D(c_1, c_i)
    end
    for i = 2 to n − 1 do
        for j = i + 1 to n − 1 do
            if |D_{c_1,c_i} − D_{c_1,c_j}| > γ then
                D_{c_i,c_j} = γ + 1
            else
                D_{c_i,c_j} = D(c_i, c_j)
            end
        end
    end
    return D_{c_i,c_j}, ∀i, j.
end
```

This modified distance matrix is then can be used in any proximity-based clustering algorithm. The algorithm changing the distance matrix is presented in Algorithm 6. The main idea of the algorithm is to first setting the distances between all must-linked (cannot-linked) points to be zero (a very large distance), so that they will be assigned to the same cluster (different clusters) by the clustering algorithm. Then, the algorithm will propagate constraints, i.e. if two points x_i and x_j are very close together, then points that are very close to x_i must be also close to x_j, or if two points x_i and x_j are very far from each other, then points that are very close to x_i must be very far from x_j. In detail, the step 2 sets all distances between points in must-link constraints to zero. Although this step tries to bring all must-link points close together, it ruins the triangle inequality of a metric. Therefore, the next step corrects this by assigning the distances between all points as the shortest paths between them (require the complexity of $O(N^3)$). This step can be considered as the step of propagating the must-link constraints. Finally, the last step forces the cannot-link points to have the distance of the longest distance plus one. This step ensures the cannot-link points have the longest distance and will be assigned to different clusters by a clustering algorithm but again, it destroys the triangle inequality of a metric. The authors argue that instead of restoring explicitly the triangle inequality, using a suitable proximity-based clustering algorithm (e.g. complete-linkage) can propagate the cannot-link constraints or still implicitly restore the triangle inequality when the clustering algorithm performs a merge. For example, given three data points x_a, x_b, x_c and the complete-linkage agglomerative clustering algorithm is used. If x_a is cannot-linked to x_b, and assume there is a violation of the triangle inequality $D_{a,b} = \max_{i,j} D_{i,j} > D_{a,c} + D_{c,b}$. Then merging cluster $\{x_a\}$ and $\{x_c\}$ (resulting in $\{x_a, x_c\}$) will also imply that $\{x_c\}$ cannot be merged with $\{x_b\}$ (because the distance between two clusters $\{x_a, x_c\}$ and $\{x_b\}$ is the distance of two furthest points in two clusters). Therefore, implicitly $D_{c,b} = \max_{i,j} D_{i,j} = D_{a,b}$, and hence $D_{a,b} < D_{a,c} + D_{c,b}$.

Algorithm 6: Changing distance matrix

Input : a set of must-link constraints $C_=$, and cannot-link constraints $C_{\neq}$, a set of data points X

Output: The modified distance matrix $D_{i,j}, \forall x_i, x_j$.

begin

1. Initialize $D_{i,j} = D_{j,i} = D(x_i, x_j)$.
2. $\forall c_=(x_i, x_j) \in C_= : D_{i,j} = D_{j,i} = 0$.
3. $\forall x_i, x_j . D_{i,j} = D_{j,i} = ShortestPath(x_i, x_j)$ using D.
4. $\forall c_{\neq}(x_i, x_j) \in C_{\neq} : D_{i,j} = D_{j,i} = \max_{a,b}(D_{a,b}) + 1$.

end

The rest part of this section will present the partitioning clustering techniques. In stead of providing a tree of clusters as the hierarchical clustering, the partitioning clustering (PC) splits the set of data objects into K disjoint clusters where K is usually provided by the user. One of the techniques which forces the well-known PC algorithm - KMEANS satisfy the constraints is to modify the assignment step of KMEANS so that only membership assignments without violating the constraints are allowed. This technique was introduced by Wagstaff et al. [51] and the algorithm was named as *COP-KMEANS*. The pseudo-code of *COP-KMEANS* is illustrated in Algorithm 7. *COP-KMEANS* is the standard *KMEANS* with the modification of the assignment step. In the assignment step, each data object d_i is assigned to the nearest cluster C_j only if this membership assignment does not violate the must-link and cannot-link constraints. The main drawback of *COP-KMEANS* is that it can fail to find a satisfying solution even if that solution exists. This comes from the greedy property of *KMEANS* when searching for the nearest clusters, and there is no backtracking mechanism. Also, *all con-*

straints must be satisfied by the clustering solutions, therefore this requires the constraints must be noiseless but this condition is rarely hold in practice.

Algorithm 7: COP-KMEANS

Input : data set $X = \{x_i\}_{i=1}^N$, number of cluster K, must-link constraints $C_= \subseteq X \times X$, cannot-link constraints $C_{\neq} \subseteq X \times X$

Output: Clusters $C_1, ..., C_K$

begin
- Initialize cluster centers $\mu_1, ..., \mu_K$.
- **repeat**
  - **for** $x_i \in X$ **do**
    - assign x_i to the nearest cluster C_j such that:

$$\nexists x_= : (x_i, x_=) \in C_=, x_= \notin C_j$$

and

$$\nexists x_{\neq} : (x_i, x_{\neq}) \in C_{\neq}, x_{\neq} \in C_j$$

    - **if** *no such cluster* C_j *exists* **then**
      - Failed and **return** $\{\}$.
    - **end**
  - **end**
  - **for** *each cluster* C_j **do**
    - update its center μ_j as the mean of all data objects $x_i \in C_j$.
  - **end**
- **until** *convergence* ;
- **return** $\{C_1, ..., C_K\}$.

end

In order to overcome the hard constraint limitation of *COP-KMEANS*, the next two *KMEANS* based algorithms *CVQE* [14] and *LCVQE* [37] allow some constraints unsatisfied (soft constraints) by adding constraint violation costs into the well-known objective function Vector Quantization Error

(VQE) of *KMEANS*. The VQE objective function is defined as:

$$VQE = \sum_{j=1}^{K} VQE_j \tag{1}$$

$$VQE_j = \sum_{x_a \in C_j} D(x_a, \mu_j)^2 \tag{2}$$

where $D(x_j, c_i)$ is the distance between a data point x_j and the nearest center c_i. The *CVQE* algorithm add the violation cost into *VQE* as follows:

$$CVQE = \sum_{j=1}^{K} CVQE_j \tag{3}$$

$$\begin{aligned} CVQE_j = & \frac{1}{2} \sum_{x_a \in C_j} D(x_a, \mu_j)^2 \\ & + \frac{1}{2} \sum_{x_a \in C_j, (x_a, x_b) \in C_=, y_a \neq y_b} D(\mu_{y_a}, \mu_{y_b})^2 \\ & + \frac{1}{2} \sum_{x_a \in C_j, (x_a, x_b) \in C_{\neq}, y_a = y_b} D(\mu_{y_a}, \mu_{h(y_b)})^2 \end{aligned} \tag{4}$$

where y_i is the cluster index which the data point x_i is assigned to, and $h(y_i)$ returns the index of the nearest cluster center (other than y_i) to the cluster y_i center. The first term of *CVQE* is *VQE*. The second term is used to penalize the must-link constraints. When a must-link constraint is violated, the penalty of the distance between two nearest cluster centers of these two points is added to the objective function. Similarly, the violation cost of cannot-link constraints contributes to the objective function *CVQE* through the third term. The cost of violating a cannot-link constraint is computed as the distance between the cluster center that these two points are assigned to and the cluster center nearest to this cluster center. For each pair of points in the constraints, the *CVQE* objective value is calculated for each combination of cluster assignments, and the cluster assignments which minimally increase the *CVQE* objective value is selected. In the *KMEANS* framework, the cluster assignment rules based on the *CVQE* objective funciton are as follows:

$$\forall x_i \notin C_= \cup C_{\neq} : y_a = \operatorname*{argmin}_j D(x_i, \mu_j)^2 \tag{5}$$

$$\forall (x_a, x_b) \in C_= : (y_a, y_b) = \operatorname*{argmin}_{i,j} D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \mathbf{1}[i \neq j] D(\mu_i, \mu_j)^2 \tag{6}$$

$$\forall (x_a, x_b) \in C_{\neq} : (y_a, y_b) = \operatorname*{argmin}_{i,j} D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \mathbf{1}[i = j] D(\mu_i, \mu_{h(j)})^2 \tag{7}$$

where $\mathbf{1}$ is the indicator function with $\mathbf{1}[true] = 1, \mathbf{1}[false] = 0$. And the cluster center μ_j is updated as:

$$\mu_j = \frac{\sum\limits_{x_i \in C_j} [x_i + \sum\limits_{(x_i, x_a) \in C_=, y_i \neq y_a} \mu_{y_a} + \sum\limits_{(x_i, x_a) \in C_{\neq}, y_i = y_a} \mu_{h(y_a)}]}{|\mu_j| + \sum\limits_{(x_i, x_a) \in C_=, y_i \neq y_a} 1 + \sum\limits_{(x_i, x_a) \in C_{\neq}, y_i = y_a} 1} \tag{8}$$

The idea of the cluster center update rule is that if a must-link constraint of two points x_i, x_a is violated then the center of the cluster containing the point x_i in that constraint is moved towards the center of the other point x_a. Similarly, when a cannot-link constraint between two points x_i, x_a is violated, the cluster center of these two points is moved towards the nearest cluster center of the current cluster center. The most expensive step in the *CVQE* algorithm is the cluster assignment step as the rules in Equ. 5, 6, 7. Especially, the assignment rules for the must-link and cannot-link pairs in Equ. 6, 7 require $O(K^2)$ complexity.

The *LCVQE* [37] modifies the *CVQE* objective function as in Equ. 10 by changing the penalty of violating a must-link constraint $c_=(x_a, x_b)$ to be the distance from second point x_b (assume the second point is the violated point) to the center of the cluster where the first point x_a is assigned to, and the penalty of violating a cannot-link constraint $c_{\neq}(x_a, x_b)$ to be the distance from the farthest point (with respect to the center of the cluster where x_a, x_b

belong to) to another nearest center.

$$LCVQE = \sum_{j=1}^{K} LCVQE_j \tag{9}$$

$$\begin{aligned} LCVQE_j = &\frac{1}{2} \sum_{x_a \in C_j} D(x_a, \mu_j)^2 \\ &+ \frac{1}{2} \sum_{(x_a,x_b) \in C_=, y_a \neq y_b, y_a = j} D(x_b, \mu_j)^2 \\ &+ \frac{1}{2} \sum_{(x_a,x_b) \in C_{\neq}, y_a = y_b, D(x_a,\mu_{y_a}) < D(x_b, \mu_{y_b}), j = h'(x_b)} D(x_b, \mu_j)^2 \end{aligned} \tag{10}$$

where $h'(x_b)$ returns the index of the the nearest center to x_b, other than y_b. $LCVQE$ improves $CVQE$ by not computing all possible K^2 combination assignments but only at most three reasonable assignments as shown in the assignment rules of Equ. 11, 12, 13.

$$\forall x_i \notin C_= \cup C_{\neq} : y_a = \operatorname*{argmin}_j D(x_i, \mu_j)^2 \tag{11}$$

$$\begin{aligned} \forall (x_a, x_b) \in C_= : (y_a, y_b) = &\operatorname*{argmin}_{[i=g(x_a), j=g(x_b)] \vee [i=j=g(x_a)] \vee [i=j=g(x_b)]} D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 \\ &+ \mathbf{1}[i \neq j] \frac{1}{2} (D(x_a, \mu_j)^2 + D(x_b, \mu_i)^2) \end{aligned} \tag{12}$$

$$\begin{aligned} \forall (x_a, x_b) \in C_{\neq} : (y_a, y_b) = &\operatorname*{argmin}_{[i=g(x_a), j=g(x_b), i \neq j] \vee [i=g(x_a), j=g(x_b), i=j, D(x_a,\mu_i) < D(x_b,\mu_j)]} \\ &D(x_a, \mu_i)^2 + D(x_b, \mu_j)^2 + \mathbf{1}[i=j] D(x_b, \mu_{g'(x_b)})^2 \end{aligned} \tag{13}$$

where $g(x_a)$ returns the index of the nearest center to the point x_a and $g'(x_a)$ returns the index of the nearest center to the point x_a, other than $g(x_a)$. The assignment rules of $LCVQE$ can be interpreted as follows. A must-link pair (x_a, x_b) can be assigned to: a) the different nearest clusters of x_a and x_b, b) the nearest cluster of x_a, or c) the nearest cluster of x_b, based on the penalty in each case. A cannot-link pair (x_a, x_b) can be assigned to: a) the different nearest clusters of x_a and x_b, or b) the same nearest cluster of both x_a and x_b, based on the penalty in each case. Finally, the center update rule of $LCVQE$ is as follows:

$$\mu_j = \frac{\sum_{x_i \in C_j} x_i + \sum_{(x_a,x_b) \in C_=, y_a \neq y_b, y_a = j} x_b + \sum_{(x_a,x_b) \in C_{\neq}, y_a = y_b, g'(x_b) = j, D(x_a,\mu_{y_a}) < D(x_b,\mu_{y_b})} x_b}{|\mu_j| + \sum_{(x_a,x_b) \in C_=, y_a \neq y_b, y_a = j} 1 + \sum_{(x_a,x_b) \in C_{\neq}, y_a = y_b, g'(x_b) = j, D(x_a,\mu_{y_a}) < D(x_b,\mu_{y_b})} 1} \tag{14}$$

Another KMEANS-based algorithm which also allows soft constraints, called *PCKMEANS*, is introduced by Basu et al. [5]. *PCKMEANS* is an extension of *KMEANS* to take into account constraints, and the Vector Quantization Error objective function is replaced by the following objective function:

$$J_{pckm} = \frac{1}{2}\sum_{h=1}^{K}\sum_{x_i \in C_h} ||x_i - \mu_h||^2 + \sum_{(x_i,x_j)\in C_=} w_{ij}\mathbf{1}[y_i \neq y_j] + \sum_{(x_i,x_j)\in C_{\neq}} w_{ij}\mathbf{1}[y_i = y_j] \tag{15}$$

where w_{ij} is the weight of the pair constraint (x_i, x_j), and other notations have the same meaning as the above notations. The first term of J_{pckm} is the Vector Quantization Error term, the second and third terms are the penalties of the constraint violation. *PCKMEANS* is represented in Algorithm 8. The first phase of the algorithm is to initialize the cluster centroids based on the constraints. From the original constraints, other constraints can be inferred through the transitivity of must-link constraints, $\{(x_1, x_2), (x_2, x_3)\} \subseteq C_= \Rightarrow (x_1, x_3) \in C_=$ and the entailment of cannot-link and must-link constraints, $(x_1, x_2) \in C_=, (x_2, x_3) \in C_{\neq} \Rightarrow (x_1, x_3) \in C_{\neq}$. The new set of constraints will be used to build the neighbourhoods. Each neighbourhood N_p is the set of all points in which two arbitrary points x_i, x_j have a must-link constraint. The second phase of the algorithm is to cluster the data points by the KMEANS algorithm with the objective function J_{pckm}. From the experimental results in the paper, the performance of *PCKMEANS* is approximately the same as the performance of *Seeded-KMEANS* (Algorithm 1) and *Constrained KMEANS* (Algorithm 2) [4]. In addition, *PCKMEANS* is a KMEANS-based algorithm, therefore it inherits the disadvantages of *KMEANS* like the local minima problem, empty clusters, etc.

In stead of incorporating the constraints into KMEANS like *COP-KMEANS*, Shental et al. [44] propose methods for integrating the constraints into the Gaussian Mixture Models (GMMs) under the Expectation Maximization (EM) framework. Due to the higher integration-complexity of cannot-link constraints, only a generalized EM algorithm using a Markov network is proposed for handling cannot-link constraints whereas a closed form EM algorithm is successfully obtained for must-link constraints. For both algorithms, the E-step of the EM algorithm is modified to compute the expectation of the complete data log-likelihood over *only cluster assignments complying strictly the given constraints*, instead of over all possible assignments as in the standard EM algorithm. From the experimental results using the datasets in the UCI repository, these algorithms are shown to outperform the constrained KMEANS *COP-KMEANS* (presented above). Another in-

Algorithm 8: PCKMEANS

Input : dataset X, set of must-link constraints $C_=$, set of cannot-link constraints $C_{\neq}$, number of clusters K, constraint weights $\{w_{ij}\}_{ij}$

Output: K clusters $\{C_1, ..., C_K\}$ such that the objective function J_{pckm} is minimized

begin

1. Initialize clusters:

   1a. create the λ neighbourhoods $\{N_p\}_{p=1}^{\lambda}$ from the constraints.

   1b. sort the indices in the decreasing size of N_p.

   1c. initialize cluster centroids:

   if $\lambda \geq K$ **then**

   initialize $\{\mu_h^{(0)}\}_{h=1}^{K}$ with centroids of $\{N_p\}_{p=1}^{\lambda}$

   else

   initialize $\{\mu_h^{(0)}\}_{h=1}^{\lambda}$ with centroids of $\{N_p\}_{p=1}^{\lambda}$

   if $\exists$ *point* x *cannot-linked to all neighbourhoods* $\{N_p\}_{p=1}^{\lambda}$ **then**

   initialize $\mu_{\lambda+1}^{(0)}$ with x

   end

   end

   Initialize remaining cluster centroids randomly.

2. Cluster:

repeat

2a. Assign cluster: assign each data point x_i to the cluster h^*:

$$h^* = \underset{h}{\mathrm{argmin}}\left(\frac{1}{2}||x_i - \mu_h^{(t)}||^2 + \sum_{(x_i,x_j)\in C_=} w_{ij}\mathbf{1}[y_i \neq y_j] + \sum_{(x_i,x_j)\in C_{\neq}} w_{ij}\mathbf{1}[y_i = y_j]\right)$$

2b. Estimate cluster centroids:

$$\mu_h^{(t+1)} = \frac{1}{|C_h^{(t+1)}|} \sum_{x_i \in C_h^{(t+1)}} x_i$$

2c. $t = t + 1$

until *convergence* ;

end

teresting result observed from the empirical results is that the performance improvement is mostly contributed by the must-link constraints. However, like *COP-KMEANS*, these two algorithms only search for the cluster assignments satisfied all constraints, and therefore these two algorithms are not suitable in the case where the constraints carry uncertainty or they are conflicting to each other. Another disadvantage is that these two algorithms suppose that the data distribution in each cluster is a Gaussian distribution which is not always true in practice and hence the performance of these two algorithms can be poor when this condition is not hold.

In order to overcome the above limitation, Lu et al. [33] introduce an extended EM algorithm, called *Penalized Probabilistic Clustering (PPC)*, that integrates pairwise constraints into the GMMs under the EM algorithm to adjust the prior distributions through a weighting function. In detail, denote $X = \{x_i\}, i = 1, ..., N$ the dataset with the latent cluster assignments $Z = \{z_{x_i}\}, i = 1, ..., N$ where $z_{x_i} \in [1, ..., K]$ and K is the number of clusters. The weighting function $g(Z, C, W)$ has large values when the assignment Z is consistent with the given pairwise constraints C and low values when Z conflicts with the constraints C. Z is parameterized by the weights W of constraints provided by users. Usually, a must-link/cannot-link constraint between two data object x_i, x_j has the weight $w(x_i, x_j) > 0$ or $w(x_i, x_j) < 0$, respectively. And the absolute value $|w(x_i, x_j)|$ presents the importance of this constraint. For example, a must-link constraint of x_i, x_j with the weight $w(x_i, x_j) = \infty$ if the resulting clustering assignment is forced to satisfy this constraint. Then the *penalized* prior distribution $P_p(Z|\Theta, C, W)$ of the latent cluster assignments Z given the configuration Θ of the GMMs and the constraints C is defined as proportional to the product of the original prior distribution $P(Z|\Theta)$ and the weighting factor $g(Z, C, W)$:

$$\begin{aligned} P_p(Z|\Theta, C, W) &= \frac{P(Z|\Theta)g(Z, C, W)}{\sum\limits_{Z} P(Z|\Theta)g(Z, C, W)} \\ &= \frac{1}{\Omega} P(Z|\Theta)g(Z, C, W) \end{aligned} \tag{16}$$

And because the incomplete data likelihood $P(X|Z, \Theta)$, given a specific cluster assignment Z, is independent of the constraints C and constraint weights W:

$$P(X, Z|\Theta, C, W) = P(X|Z, \Theta)P(Z|\Theta, C, W)$$

therefore, the *penalized* complete data likelihood is written as:

$$\begin{aligned} P_p(X, Z|\Theta, C, W) &= P(X|Z, \Theta)P_p(Z|\Theta, C, W) \\ &= \frac{1}{\Omega}P(X|Z, \Theta)P(Z|\Theta)g(Z, C, W) \\ &= \frac{1}{\Omega}P(X, Z|\Theta)g(Z, C, W) \end{aligned}$$

where $P(X, Z|\Theta)$ is the complete data likelihood in a standard GMM. And the expectation (E-step) and maximization step (M-step) of the EM algorithm which is used to maximize the expected value of the complete data log likelihood with respect to Θ at step t are as follows:

$$\begin{aligned} \text{E-step: } & \text{Evaluate } P_p(Z|\Theta^{(t-1)}, C, W) \\ \text{M-step: } & \Theta^{(t)} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^{(t-1)}) \end{aligned}$$

where

$$Q(\Theta, \Theta^{(t-1)}) = \sum_{Z} P_p(Z|\Theta^{(t-1)}, C, W) log P_p(X, Z|\Theta, C, W)$$

As can be seen from Equ. 16, if $\forall Z.\ g(Z, C, W) = 1$, *PPC* reduces to the standard EM algorithm. If $\forall Z.\ Z$ satisfies constraints C then $g(Z, C, W) = 1$ and $\forall Z.\ Z$ does not satisfy constraints C then $g(Z, C, W) = 0$, *PPC* reduces to the case of *hard constraints* (constraints must be satisfied in the resulting cluster assignments) proposed by Shental et al. [44] (presented above). In other cases, the *PPC* framework allows both *hard constraints* and *soft constraints* (constraints can be missed in the resulting cluster assignments) by setting a suitable weight for each constraint in C for the weighting function $g(Z, C, W)$. Therefore, the constraints can be specified even when they are noisy. Despite of the advantage of flexibility, this framework also suffers many disadvantages. First, like other GMM based algorithms, *PPC* assumes that data objects in each cluster can be approximated by a Gaussian distribution. However, this condition is often unsatisfied in practice and can lead to poor results. Second, the weighting function $g(Z, C, W)$ are based on the weights of constraints provided by users, and tuning these weights to have good results is a non-trivial task. Also, the *PPC* framework is not a closed form EM, therefore the update of the prior probability is approximated by different techniques described in detail in the paper.

Another probabilistic framework based on Hidden Markov Random Fields (HMRFs) which combines both the constraint-based and distance-based approach is proposed by Basu et al. [6, 3]. In this framework, the HMRF model consists of the following components:

- An observable set $X = \{x_i\}_{i=1}^{M}$ of the M given data points X.
- An hidden set $Y = \{y_i\}_{i=1}^{M}$ of the cluster labels of data points, and $y_i \in \{h\}_{h=1}^{K}$ where K is the number of clusters.
- An hidden set $\Theta = \{\Theta_i\}_{i=1}^{K}$ of the generative model parameters of clusters.
- An observable set of constraint variables $C = \{c_{ij}\}_{1 \leq i,j \leq M, i \neq j}$ where $c_{ij} = 1$ or $c_{ij} = -1$ means existing a must-link constraint $c_{=}(x_i, x_j)$ or a cannot-link constraint $c_{\neq}(x_i, x_j)$ between (x_i, x_j), respectively and $c_{ij} = 0$ if there is no constraint on the pair (x_i, x_j).

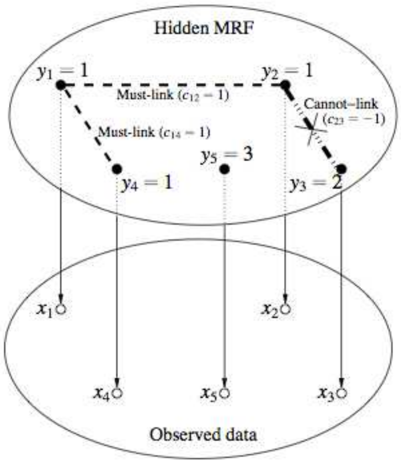


Figure 5: An example of modelling pairwise constraints by a HMRF

Fig. 5 shows an example of modelling constraints by the HMRF extracted from [3]. In this example, there are 5 data points $\{x_1, .., x_5\}$ and the goal is to split them into 3 clusters, therefore the cluster labels y_i can only be a value in $\{1, 2, 3\}$. Also, there are 2 must-link constraints of (x_1, x_4), (x_1, x_2) and 1 cannot-link constraint of (x_2, x_3) and the corresponding constraint variables are $c_{14} = 1, c_{12} = 1, c_{23} = -1$. For other pairs (x_i, x_j) without constraints, the constraint variables $c_{ij} = 0$. A cluster assignment satisfied the constraints is $y_1 = 1, y_2 = 1, y_3 = 2, y_4 = 1, y_5 = 3$. Denote the neighbourhood N_i the set of neighbours of y_i and $N = \{N_i\}_{1 \leq i \leq M}$. In the case of pairwise constraints, N_i is the set of cluster labels y_j such that there exists a must-link or cannot-link constraint between x_i and x_j. Formally, N_i is defined as:

$$N_i = \{y_j | c_{=}(x_i, x_j) \text{ or } c_{\neq}(x_i, x_j)\} \tag{17}$$

And according to the Hammersley-Clifford theorem [27], the prior probability of a label configuration Y follows the Gibbs distribution [20]:

$$P(Y|\Theta, C) = \frac{1}{Z}exp(-\nu(Y)) = \frac{1}{Z}exp(-\sum_{N_i \in N} \nu_{N_i}(Y)) \tag{18}$$

where Z is the normalizing term, $\nu(Y)$ is the overall label configuration potential function that can be decomposed into the functions $\nu_{N_i}(Y)$ which are the potentials for all neighbourhoods N_i in the label configuration Y. And because the neighbourhoods N_i are defined based on pairwise constraints in C, the prior distribution of a configuration Y in Equ. 18 can be rewritten as:

$$P(Y|\Theta, C) = \frac{1}{Z}exp(-\sum_{i,j} \nu(i,j)) \tag{19}$$

where

$$\nu(i,j) = \begin{cases} f_{=}(x_i, x_j) & \text{if } c_{=}(x_i, x_j) \\ f_{\neq}(x_i, x_j) & \text{if } c_{\neq}(x_i, x_j) \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

Here, $f_{=}(x_i, x_j)$ and $f_{\neq}(x_i, x_j)$ are the non-negative functions which penalize the violation of must-link or cannot-link constraints, respectively. Besides, each data point x_i is assumed to be drawn i.i.d from the conditional probability $P(x_i|y_i, \Theta)$, hence:

$$P(X|Y, \Theta, C) = P(X|Y, \Theta) = \prod_{i=1}^{M} p(x_i|y_i, \Theta) \tag{21}$$

From Equ. 19 and Equ. 21, the joint probability $P(X, Y, \Theta|C)$ can be written as:

$$\begin{aligned} P(X, Y, \Theta|C) &= P(\Theta|C)\ P(Y|\Theta, C)\ P(X|Y, \Theta, C) \\ &= P(\Theta)\ \frac{1}{Z}exp(-\sum_{c_{ij} \in C} \nu(i,j))\ \prod_{i=1}^{M} p(x_i|y_i, \Theta) \end{aligned} \tag{22}$$

where the prior distribution $P(\Theta|C)$ of the parameters Θ is independent of C, i.e. $P(\Theta|C) = P(\Theta)$. If the condition probability $p(x_i|y_i, \Theta)$ is restricted to the exponential forms, then:

$$p(x_i|y_i, \Theta) = \frac{1}{Z_\Theta}exp(-D(x_i, \mu_{y_i})) \tag{23}$$

where $D(x_i, \mu_{yi})$ is the distance between the data point x_i and the mean μ_{yi} of the points in a cluster with the same label y_i. Substituting Equ. 23 into Equ. 22 and taking negative logarithms gives the following cluster objective function:

$$J_{\text{hmrf-kmeans}} = \sum_{x_i \in X} D(x_i, \mu_{yi}) + \sum_{c_{ij} \in C} \nu(i,j) - logP(\Theta) + logZ + logZ_{\Theta} \quad (24)$$

$$J_{\text{hmrf-kmeans}} = \sum_{x_i \in X} D(x_i, \mu_{yi}) + \sum_{c_{=}(x_i,x_j)} f_{=}(x_i, x_j) + \sum_{c_{\neq}(x_i,x_j)} f_{\neq}(x_i, x_j)$$
$$- logP(\Theta) + logZ + logZ_{\Theta} \quad (25)$$

Minimizing this function with respect to Y and Θ is equivalent to maximizing the joint probability $P(X, Y, \Theta|C)$ in Equ. 22. In a simple form, the must-link penalty function can be $f_{=}(x_i, x_j) = w_{ij}\mathbf{1}[y_i \neq y_j]$ where w_{ij} is the weight of violating a must-link constraint $c_{=}(x_i, x_j)$, and $\mathbf{1}$ is the indicator function ($\mathbf{1}[true] = 1, \mathbf{1}[false] = 0$). However, this function only penalizes with the cost of w_{ij} whenever a must-link constraint $c_{=}(x_i, x_j)$ is violated but without considering the distance between two points x_i and x_j. In other words, if two must-link constraints $c_{i1,j1} = 1$ (i.e. $c_{=}(x_{i1}, x_{j1})$) and $c_{i2,j2} = 1$ (i.e. $c_{=}(x_{i2}, x_{j2})$) have the same violation weight $w_{i1,j1} = w_{i2,j2}$, and $D(x_{i1}, x_{j1}) > D(x_{i2}, x_{j2})$, the cost of violating the must-link constraint $c_{i1,j1}$ of distant points should be higher than of violating the must-link constraint $c_{i2,j2}$ of nearby points. Because if two must-linked points are close to each other and the violation happens, it means that the distance function already works correctly, the violation problem mostly comes from the clustering algorithm. In contrast, if two must-linked points are far from each other and the violation happens, it means that the violation problem mostly causes by the distance function, and therefore in this case the distance function must be modified to bring those points closer to each other. From this point of view, the must-link penalty function is changed as:

$$f_{=} = w_{ij}\varphi_D(x_i, x_j)\mathbf{1}[y_i \neq y_j] \quad (26)$$

where $\varphi_D(x_i, x_j)$ is the penalty scaling function and is a monotonically increasing function of the distance between x_i and x_j according to the distance function D. Similarly, the cost of violating a cannot-link constraint of two near points should be higher than of two far points according to the current distance function. The cannot-link penalty function is chosen as:

$$f_{\neq} = \overline{w}_{ij}(\varphi_{Dmax} - \varphi_D(x_i, x_j))\mathbf{1}[y_i = y_j] \quad (27)$$

where φ_{Dmax} is the maximum value of the scaling function φ_D for the dataset. Substituting Equ. 26, and Equ. 27 into Equ. 25 produces the cluster objective function parameterized by the distance function D together with Y and Θ:

$$\begin{aligned} J_{\text{hmrf-kmeans}} = & \sum_{x_i \in X} D(x_i, \mu_{yi}) + \sum_{c_=(x_i,x_j)} w_{ij}\varphi_D(x_i,x_j)\mathbf{1}[y_i \neq y_j] \\ & + \sum_{c_{\neq}(x_i,x_j)} \overline{w}_{ij}(\varphi_{Dmax} - \varphi_D(x_i,x_j))\mathbf{1}[y_i = y_j] \\ & - logP(\Theta) + logZ + logZ_{\Theta} \end{aligned} \tag{28}$$

If the distance function D is parameterized by parameters Θ_D, then the goal now is to minimize the objective function $J_{\text{hmrf-kmeans}}$ with respect to the cluster parameters Θ, the cluster assignment Y and the distance function parameters Θ_D. The EM algorithm used to optimize this objective function is shown in Algorithm 9. The *HMRF-KMEANS* framework in Algorithm 9 is shown to converge to a local minimum of $J_{\text{hmrf-kmeans}}$ in [6]. In addition, the *PKM* (pairwise constraint *KMEANS*), *MKM* (metric learning *KMEANS*), and *MPKM* (metric pairwise constraint *KMEANS*) algorithm in [9] are just special cases of this framework. Although this framework is flexible, computing the global optimal solutions for Θ, Θ_D, Y in the E and M steps is a non-trivial task, hence many approximation techniques have been used to estimate only the local optimal solutions and this can result in a poor local optimum. Also, the cluster models must be approximately in the exponential form to guarantee a good result.

Distanced-Based Clustering

In this approach, only the distance function is adjusted based on the constraints such that the must-link points are placed near each other and the cannot-link points must be far from each other.

Xing et al. [52] formalize the distance metric learning problem as an optimization problem. The goal is to minimize the objective function which is the sum of distances of pairs in the must-link constraints, subject to the condition that the distances of pairs in the cannot-link constraints are greater than a constant. The distance function $d_A(x_1, x_2)$ between two points x_1, x_2 used in the objective function is a Mahanabolis metric parameterized by a

Algorithm 9: HMRF-KMEANS

Input : Set of data points $X = x_i{}_{i=1}^{M}$, number of clusters K, set of constraints C, distance measure D (parameterized by Θ_D), constraint violation weights W.

Output: K disjoint clusters (represented by cluster assignment Y) such that the objective function $J_{\text{hmrf-kmeans}}$ is minimized.

begin

Initialize the cluster parameters Θ.

repeat

E-step:

- fix Θ, and Θ_D: minimize $J_{\text{hmrf-kmeans}}$ with respect to Y.

M-step:

1. fix Y, and Θ_D: minimize $J_{\text{hmrf-kmeans}}$ with respect to Θ.
2. fix Y, and Θ: minimize $J_{\text{hmrf-kmeans}}$ with respect to Θ_D.

until *convergence* ;

end

matrix A. In mathematical form, the optimization problem is:

$$\min_{A} \sum_{(x_i,x_j)\in C_=} d_A^2(x_i - x_j) \tag{29}$$

$$\text{s.t.} \sum_{(x_i,x_j)\in C_{\neq}} d_A(x_i - x_j) \geq 1, \tag{30}$$

$$A \succeq 0 \tag{31}$$

where

$$d_A(x_i - x_j) = (x_i - x_j)^T A(x_i - x_j) \tag{32}$$

Minimizing the objective function in Equation 29 is equivalent to search for a matrix A such that the distances between the must-link points are as small as possible but it still guarantees that the distances between the cannot-link points are larger than a constant. In order to ensure that d_A is a valid metric (satisfying the non-negativity and triangle inequality properties), the matrix A must be positive semi-definite. Also, the optimization problem in Equation 29 is convex and differentiable, therefore it is possible to derive efficient algorithms to find the global optimum. When A is a diagonal matrix, an alternative optimization problem is proposed:

$$\min_A \sum_{(x_i,x_j)\in C_=} d_A^2(x_i,x_j) - log \sum_{(x_i,x_j)\in C_{\neq}} d_A(x_i,x_j) \tag{33}$$

$$\text{s.t. } A \succeq 0 \tag{34}$$

The Newton-Raphson method can be applied in this case to find the global optimum. However, when the matrix A is not diagonal, the Newton-Raphson method is very expensive (because of the computation complexity $O(n^6)$ of the inverse of the Hessian matrix for n^2 parameters), and therefore another equivalent problem is proposed by the authors:

$$\max_A g(A) = \sum_{(x_i,x_j)\in C_{\neq}} d_A(x_i,x_j) \tag{35}$$

$$\text{s.t. } f(A) = \sum_{(x_i,x_j)\in C_=} d_A^2(x_i,x_j) \leq 1 \tag{36}$$

$$A \succeq 0 \tag{37}$$

A gradient ascent combined iterative projection algorithm is used to solve the optimization problem in Equ 35. This algorithm is represented in Algorithm 10. The projection steps of the algorithm are used to ensure that the

Algorithm 10: Gradient ascent combined iterative projection algorithm.

begin
    $C_1 = \{A : \sum_{(x_i,x_j)\in C_=} d_A^2(x_i,x_j) \leq 1\}$
    $C_2 = \{A : A \succeq 0\}$
    repeat
        repeat
projection 1 $A = \text{argmin}_{A'}\{||A' - A||_F : A' \in C_1\}$
projection 2 $A = \text{argmin}_{A'}\{||A' - A||_F : A' \in C_2\}$
            // where $||.||_F$ is the Frobenius norm on matrices,
            // $||M||_F = (\sum_i \sum_j M_{ij}^2)^{1/2}$
        until A *converges* ;
gradient step $A = A + \alpha\nabla_A g(A)$
    until *convergence* ;
end

constraints in Equ. 36 and 37 hold. And the objective function $g(A)$ in Equ.

35 is optimized by the gradient step. The reason why the authors formalize the original problem as in Equ. 35, 36, 37 is because the projection steps can be done inexpensively (the complexity of $O(n^2)$ for the projection on C_1 and the complexity for the projection on C_2 is the complexity of decomposing $A = X^T \Lambda X$).

Another distance metric learning algorithm, called *DistBoost*, which is an extension of the Adaboost algorithm [41, 42] to handle unlabelled data points is proposed by Hertz et at. [28]. The distance metric learning problem is converted into a classification problem where the goal is to learn a distance function $f : X \times X \to [-1, 1]$ with X is the original dataset. A must-link/cannot-link pair will be a data point in the product space with the label of 1, or -1, respectively. For other pairs which are not in constraints, their labels are $*$. The distance function f will be the weighted sum of all weak hypothesises h_t trained at step t of the boosting scheme. At the beginning, all pairs have the same weight. Then, the weights of the misclassified pairs will be increased so that in the next iteration $t + 1$, the weak hypothesis h_{t+1} will focus to satisfy these pairs. The pseudo-code of the algorithm is illustrated in Algorithm 11. The main idea of the algorithm is to build the weak hypothesis h_t in an iteration step t from the Gaussian Mixture Models (GMM, parameterized by Θ) trained on the original data points $x_i \in X$ with weights w_i, the hidden labels $y_i \in Y$ and equivalence constraints in $C_=$ and , $C_{\neq}$. While the GMM is trained from the original dataset (using the constrained GMM-EM algorithm in [44]), the weak hypothesis h_t of the boosting scheme has the domain of the product space $X \times X$. Then the hypothesis weight α_t is computed to update the pair weights w_{ij}. And the individual point weight w_i is calculated by marginalizing over all related pair weights w_{ij}. The weights w_i of single points are incorporated into the constrained GMM-EM algorithm by modifying the original dataset such that a data point x_i with the weight of w_i will have $w_i N$ copies on the new dataset. x_i and all of its copies will form must-link constraints and therefore will be assigned to the same cluster by the constrained GMM-EM algorithm. From the experimental results, the *DistBoost* algorithm outperforms the Mahalanobis distance learning algorithm [52] (presented above), the constrained GMM-EM algorithm [44] (presented in the family of the EM based algorithms), and COP-KMEANS [51] presented in Algorithm 7.

The BoostCluster algorithm in [32] proposed by Liu et al. improves the performance of an arbitrary clustering algorithm by using pairwise constraints as side information and the previous result of that clustering algorithm to generate a new data representation of the clustering data at each iteration. The main idea of BoostCluster is to project all data objects into a subspace in which the must-link pairs are close to each other and the cannot-

Algorithm 11: The *DistBoost* algorithm.

Input : data points $X = \{x_i\}_{i=1}^N$, must-link constraints $C_{=}\{pair_t(x_i, x_j)\}_{t=1}^{ML}$, cannot-link constraints $C_{\neq}\{pair_t(x_i, x_j)\}_{t=1}^{CL}$

Output: The final hypothesis $f(x_i, x_j) = \sum_{t=1}^T \alpha_t h_t(x_i, x_j)$

begin

Set labels y_{ij} of constraints as: $y_{ij} = \begin{cases} 1 & \text{if } (x_i, x_j) \in C_{=}. \\ -1 & \text{if } (x_i, x_j) \in C_{\neq}. \\ 0 & \text{otherwise.} \end{cases}$

Initialize $w_{ij} = \frac{1}{N^2}$ for all $x_i, x_j \in X$.

for $t = 1, .., T$ **do**

Update individual point weight $w_i = \sum_j w_{ij}$.

Learn GMM parameterized by Θ on data points $x_i \in X$ with weights w_i, hidden labels $y_i \in Y$ under the constraints.

Partition X from the Maximum A Posterior assignment Y^* of points, and set:

$sign(x_i, x_j) =$

$\begin{cases} 1 & \text{if } x_i, x_j \text{ are assigned to the same cluster.} \\ -1 & \text{otherwise.} \end{cases}$

Build weak hypothesis:

$h_t(x_i, x_j) = sign(x_i, x_j) \max_{t_1} p(y_i = t_1|\Theta) \max_{t_2} p(y_j = t_2|\Theta)$.

Compute hypothesis weight: $\alpha_t = \frac{1}{2} ln \frac{1+r_t}{1-r_t}$ where

$r_t = \sum_{1 \le i,j \le N} w_{ij} h_t(x_i, x_j)$.

Update pair weight:

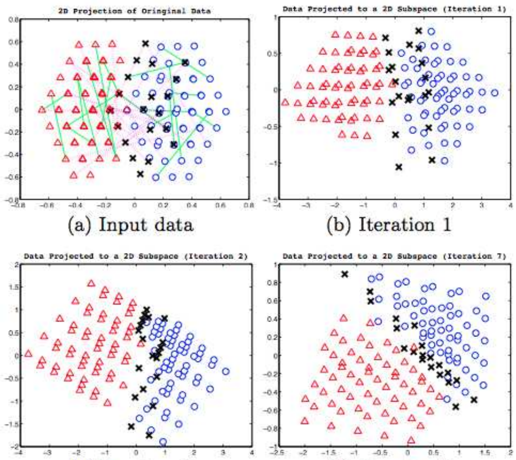
$w_{ij} = \begin{cases} w_{ij} exp(-\alpha_t y_{ij} h_t(x_i, x_j)) & \text{if } (x_i, x_j) \in C_{=}, or \in C_{\neq}. \\ w_{ij} exp(-\alpha_t) & \text{otherwise.} \end{cases}$

Normalize: $w_{ij} = w_{ij}/Z$ where $Z = \sum_{i,j} w_{ij}$.

end

return $f(x_i, x_j) = \sum_{t=1}^T \alpha_t h_t(x_i, x_j)$.

end



(a) Input data (b) Iteration 1

(c) Iteration 2 (d) Iteration 7

Figure 6: Iterative data projections of BoostClusters

link pairs are far from each other. The projected data is then used as the input to the clustering algorithm. Fig. 6 extracted from [32] illustrates this idea. The must-link and cannot-link constraints are the green solid lines and the purple dotted lines, respectively. In this dataset, there are three clusters Δ, x, o. At the beginning, these three clusters are heavily overlapped. But after each iteration, the clusters in the projected data are separated better and better. In detail, given a dataset $X = \{x_i\}_{i=1}^{N}$, a must-link constraint set $C_=$, a cannot-link constraint set $C_{\neq}$, and denote $K \in R^{N\times N}$ the kernel similarity matrix where $K_{ij} \geq 0$ is the confidence that two points x_i and x_j, the problem is to minimize the following objective function:

$$L(K) = \Big(\sum_{(x_i,x_j)\in C_=} exp(-K_{i,j})\Big)\Big(\sum_{(x_a,x_b)\in C_{\neq}} exp(K_{a,b})\Big) \tag{38}$$

The first term of the objective function is the disagreement between the kernel similarity matrix and the must-link constraints. Similarly, the second term is the inconsistency between the kernel similarity matrix and the cannot-link constraints. This objective function is minimized iteratively in a boosting style by updating the kernel similarity matrix $K^{(t)}$ at step t as follows:

$$K^{(t)} = K^{(t-1)} + \alpha^{(t)}\Delta^{(t)} \tag{39}$$

where $\alpha^{(t)} \geq 0$ is the weight combination, and $\Delta^{(t)} \in R^{N\times N}$ is the incremental kernel similarity matrix inferred from the clustering result $C^{(t)}$ of the algorithm $\mathcal{A}$ on the transformed input data $X^{(t)}$ at step t. $\Delta^{(t)}_{i,j} = 1$ if x_i and x_j are assigned in the same cluster by $C^{(t)}$, otherwise $\Delta^{(t)}_{i,j} = 0$. $\Delta^{(t)}$

can be considered as a function of the clustering result $C^{(t)}$ or a function of the input data $X^{(t)}$ at step t, given the algorithm $\mathcal{A}$, i.e. $\Delta^{(t)}$ can be written as $\Delta^{(t)} = g(C^{(t)} = g(\mathcal{A}(X^{(t)}))$. Also, assume that the input data $X^{(t)}$ at step t is the transformation of the input data $X^{(t-1)}$ in the previous step by a transformation matrix $P^{(t)}$, i.e. $X^{(t)} = (P^{(t)})^T X^{(t-1)}$. Then, the optimization problem must be solved at each iteration t is:

$$\min_{\alpha^{(t)}} \; L(K^{(t-1)} + \alpha^{(t)}\Delta^{(t)}) \tag{40}$$

$$\min_{\alpha^{(t)}} \; L(K^{(t-1)} + \alpha^{(t)} g(\mathcal{A}(X^{(t)})) \tag{41}$$

$$\min_{\alpha^{(t)}, P^{(t)}} \; L(K^{(t-1)} + \alpha^{(t)} g(\mathcal{A}((P^{(t)})^T X^{(t)})) \tag{42}$$

This optimization problem is solved by first computing the optimum project matrix $P^{(t)}$ and then searching for the optimum value of $\alpha^{(t)}$ (please refer the paper for details). The *BoostCluster* algorithm is summarized as in Algorithm 12. The *BoostCluster* algorithm is proven to converge with

---

Algorithm 12: BoostCluster

Input : Input data X, a must-link constraint set $C_=$, a cannot-link constraint set $C_{\neq}$, a clustering algorithm $\mathcal{A}$
Output: clustering result $C^{(T)}$, transformation matrix $P^{(T)}$
begin
  1. Initialize $K_{ij}^{(0)} = 0$ for all i, j.
  2. Optimize the objective function:
  for $t = 1$ *to* T **do**
    Fix $\alpha^{(t)}$, optimize Equ. 42 with respect to $P^{(t)}$.
    Fix $P^{(t)}$ with the optimum value found in the previous step, optimize Equ. 42 with respect to $\alpha^{(t)}$.
    Transform input data: $X^{(t)} = (P^{(t)})^T X^{(t-1)}$.
    Run the algorithm $\mathcal{A}$ on $X^{(t)}$ to obtain a clustering $C^{(t)}$ and then compute $\Delta^{(t)}$.
    Compute $K^{(t+1)}$ as: $K^{(t+1)} = K^{(t)} + \alpha^{(t)}\Delta^{(t)}$.
  end
  return *clustering result* $C^{(T)}$, *transformation matrix* $P^{(T)}$
end

---

the exponential speed. Also, the experiments shows that the combination of *BoostCluster* with other popular algorithms like *KMEANS*, *Partitional SingleLink* [30], or *K-Way Spectral Clustering* [36] outperforms the *HMRF-KMEANS* algorithm proposed by Basu et al. [6] (presented in Section 3) in most cases.

3.2.2 Cluster-Level Constraints

In this section, the cluster-level constraints will be discussed. Some examples of cluster-level constraints are: the size of all clusters must be greater than an integer value m, the clustering result must be different from a given clustering result, etc. For many real-life applications, the balance property (all clusters have approximately the same size) of a clustering result is important, e.g. in a marketing campaign, the partitioning of customers in roughly equal size groups make the allocating of sales teams, money to each group more easily [46, 53], or in category management where one of key operations is to group the products into categories with specified sizes [39]. And a scalable framework for balanced clustering has been proposed by Banerjee et al. in [1]. In this framework, the objective is to minimize the vector quantization error in Equ. 2 of KMEANS under the constraint that the size of each cluster must be greater or equal to an integer value m. The framework is consists of three steps as in Algorithm 13. From the experimental results, the performance of the balanced clustering algorithms are slightly worse than unconstrained clustering algorithms in the case of the unbalanced data and better in the case of the balanced data. In both cases, the balanced clustering algorithms guarantee the size constraint is satisfied and result in the small size variances whereas the unconstrained clustering algorithms produce clusters with large size variances.

Algorithm 13: A scalable framework for balanced clustering

Input : Dataset X, number of clusters K, minimal cluster-size m
Output: K disjoint clusters with the minimal cluster size greater or equal to m

begin

Sampling from the given data to get a small representative subset of the data.
Clustering of the sampled data by any clustering algorithm.
Populating and refining the clusters.

- Populating: assign the remaining points to the clusters (by searching for the nearest centroids in most cases) such that the size constraint is satisfied.
- Refining: improve the objective function by re-assign points to other clusters.

end

Another approach which formalizes the balanced clustering problem as a constrained optimization problem is proposed by Demiriz et al. [18]. Let $X = \{x_i\}_{i=1}^{N}$ be the data set of N points in R^D, K be the desired number of clusters, $\{\mu_i\}_{i=1}^{K}$ be the set of cluster centers in R^D, and $T_{i,h}$ be the selection variables where $T_{i,h} = 1$ means the data point x_i is assigned to cluster h, and zero otherwise. The standard clustering problem is formalized as:

$$\begin{aligned} \min_{\mu,T} & \sum_{i=1}^{N} \sum_{h=1}^{K} T_{i,h} ||x_i - \mu_h||^2 \\ s.t. & \sum_{h=1}^{K} T_{i,h} = 1, i = 1, ..., N \\ & T_{i,h} \geq 0, i = 1, ..., N, h = 1, ..., K \end{aligned} \tag{43}$$

If the balanced constraint requires that each cluster h must contain at least τ_h data points, where $\sum_{h=1}^{K} \tau_h \leq N$, then the balanced clustering problem is formalized as follows:

$$\begin{aligned} \min_{\mu,T} & \sum_{i=1}^{N} \sum_{h=1}^{K} T_{i,h} ||x_i - \mu_h||^2 \\ s.t. & \sum_{i=1}^{N} T_{i,h} \leq \tau_h, h = 1, ..., K \\ & \sum_{h=1}^{K} T_{i,h} = 1, i = 1, ..., N, \\ & T_{i,h} \geq 0. i = 1, ..., N, h = 1, ..., K. \end{aligned} \tag{44}$$

This constrained optimization problem is shown in [18] to have an equivalent minimum cost flow (MCF) linear network optimization problem [8] with integer optimal solutions when fixing $\{\mu_i\}_{i=1}^{K}$. The balanced KMEANS algorithm for this problem is presented in Algorithm 14. From the experimental results, the balanced KMEANS performance is significantly worse than the standard KMEANS performance. This comes from the fact that the input datasets are selected such that the standard KMEANS result in many empty clusters and therefore, the size constraints force the balanced KMEANS to assign data points to the empty or the near empty of clusters. However, the cluster size variance of the standard KMEANS is much larger than the one of the balanced KMEANS. Other extensions of the formulation in Equ. 44 for other kinds of constraints are also presented in [18].

Algorithm 14: The balanced KMEANS clustering algorithm

Input : Dataset $X = \{x_i\}_{i=1}^N$, initial cluster centers $\{\mu_i^{(0)}\}_{i=1}^K$

Output: final cluster centers $\{\mu_i^{(t)}\}_{i=1}^K$, selection variable $T_{i,h}, i = 1, ..., N, h = 1, ..., K$ which is a local optimum of the optimization problem in Equ. 44.

begin

$t = -1$

repeat

$t = t + 1$

Cluster Assignment. Fix $\mu_h^{(t)}$, and let $T_{i,h}^{(t)}$ be the solution of the following optimization problem:

$$
\begin{aligned}
\min_T \sum_{i=1}^N \sum_{h=1}^K & T_{i,h}||x_i - \mu_h^{(t)}||^2 \\
s.t. \sum_{i=1}^N & T_{i,h} \le \tau_h, h = 1, ..., K \\
\sum_{h=1}^K & T_{i,h} = 1, i = 1, ..., N \\
& T_{i,h} \ge 0, i = 1, ..., N, h = 1, ..., K
\end{aligned}
$$

Cluster Update. Update $\mu_h^{(t+1)}$ as follows:

$$
\mu_h^{(t+1)} = \begin{cases} \frac{\sum_{i=1}^N T_{i,h}^{(t)} x_i}{\sum_{i=1}^N T_{i,h}^{(t)}} & \text{if } \sum_{i=1}^N T_{i,h}^{(t)} > 0 \\ \mu_h^{(t)} & \text{otherwise} \end{cases}
$$

until $\mu_h^{(t)} = \mu_h^{(t+1)}, h = 1, ...K$;

end

In all SSC algorithms discussed so far, the side information is used to guide the clustering algorithm towards the desired clustering result. In contrast, another SSC algorithm proposed by Gondek et al. [24], requires the side information is an *undesired* clustering and the goal is to find a clustering which is as much *different* as possible from this undesired clustering. The difference between two clusters C_i and C_j is measured by the variation of information $VI(C_i, C_j)$ [35] and defined as follows:

$$VI(C_i, C_j) = H(C_i) + H(C_j) - 2I(C_i, C_j) \tag{45}$$
$$\text{where } H \text{ is the entropy function}$$
$$I \text{ is the mutual-information function}$$

The variation of information $VI(C_i, C_j)$ between two clusters C_i and C_j is maximal if only if: $I(C_i, C_j; X) = I(C_i; X) + I(C_j; X)$, i.e. C_i and C_j are independent from each other. In this case, C_i and C_j are said to be *information-orthogonal*. The problem of finding a high-quality clustering which is different from a given clustering is also known as the non-redundant clustering [21, 40]. Formally, the non-redundant clustering problem given dataset X and an objective function L (usually the VQE objective function in Equ. 2) is defined as:

$$\begin{aligned} &\max_C L(X, Z) \\ &\text{s.t. } C \text{ is a valid clustering,} \\ &\quad\quad C \text{ and } Z \text{ are information-orthogonal.} \end{aligned} \tag{46}$$

In practice, the last condition is too strict and often relaxed as the variation of information of two clusters is greater than some threshold α. The application of this problem is to detect the novel clusterings or to avoid already-known clusterings (which can be trivial). The algorithm proposed by Gondek et al. [24], called *CondEns* for Conditional Ensemble clustering, consists of three steps. The first step is to partition the dataset into clusters Z^i based on the given clustering Z. Then, each cluster Z^i will be partitioned again into local clusters by a base clustering algorithm. The local clustering solution for each cluster Z^i is denoted as $\hat{C}_i$. In the second step, each local clustering $\hat{C}_i$ is extended to a global clustering C_i by assigning instances in clusters $Z^j, j \neq i$ to the local clusters of $\hat{C}_j$. Finally, the last step of the algorithm combines all global clustering solutions C_i into the target clustering C. The pseudo-code of the algorithm is illustrated in Algorithm 15. The idea of the algorithm is that by starting from the local clusters of each cluster under the undesired clustering, the target clustering will be much different from the undesired clustering. The reason is that if under the undesired clustering,

Algorithm 15: *CondEns* Algorithm

Input : Dataset $X = \{x_i\}_{i=1}^{N}$, an undesired clustering $Z : X \rightarrow \{1, .., L\}$, number of clusters K in target clustering, number of clusters K_i for each local clustering

Output: Clustering $C : X \rightarrow \{1, ..., K\}$

begin

Clustering

Let Z^i be the i-th cluster of X under the undesired clustering Z: $Z^i = \{x_j : x_j \in X, Z(x_j) = i\}$.

Apply a base clustering algorithm to each Z^i to find a local clustering $\hat{C}_i$:

$\hat{C}_i : Z^i \rightarrow \{1, ..., K_i\}, i = 1, ..., L.$

Extension

Extend each local clustering $\hat{C}_i$ to a global clustering C_i by assigning instances in $Z^j, j \neq i$ to one of the local clusters of $\hat{C}_i$:

$C_i : X \rightarrow \{1, ..., K\}, i = 1, ..., L.$

Combination

Combine clustering solutions C_i to form the target clustering:

$C = Combine(C_1, ..., C_L)$ where $C : X \rightarrow \{1, ..., K\}$.

end

all instances in a cluster Z^i are in the same cluster, then under the target clustering C, the instances of cluster Z^i are split into different clusters. In addition, the authors prove that "if the target clustering is dominant and information-orthogonal to the given clustering, then the target clustering will be among the clustering solutions handed to the combination stage". The statement requires too strict assumption which is often not hold in practice. Also, from this statement, it can be seen that the target solution should be selected from the set of potential clusterings formed at the Extension step, instead of combining them as in the Combination step. However, through the experiments, the authors claim that the combination of the potential clusterings provides a more useful clustering. Besides, a disadvantage of the algorithm is that the potential clusterings can be the local optima obtained by the base clustering algorithm and they can be inconsistent. Therefore, this can lead to the poor performance when combining them to form the target clustering.

Another algorithm which also takes into account the negative side information is called Coordinated Conditional Information Bottleneck (CCIB) [23]. CCIB is an extension of the Information Bottleneck (IB) method [47], an unsupervised method for extracting relevant structure from data and a special case of the Rate Distortion theory [43, 13, 25]. The main idea of IB is to model the structure extraction problem as a data compression problem. Given two variables X and Y, the goal of IB is to find a compact representation C parameterized by probabilities $\{p(c|x)\}$ of X such that the compact representation C preserves the information of the relevance variable Y as much as possible, or the mutual information $I(C;Y)$ between C and Y is maximized under the constraint that the information rate $I(C;X)$ between C and X is less than a threshold R. The formal representation of the IB problem is as follows:

$$\max_{p(c|x)} I(C;Y) \tag{47}$$

$$\text{s.t. } I(C;X) \leq R, \tag{48}$$

$$\forall x. \sum_c p(c|x) = 1, \text{ and } \forall x, c.\ p(c|x) \leq 0 \tag{49}$$

In the clustering problem, X is the random variable of documents, Y is the random variable of features and C is the random variable of clusters. Note that $p(y|x), p(x)$ are given or can be estimated from the dataset. And $p(y|c) = \frac{1}{p(c)} \sum_x p(x)p(c|x)p(y|x)$, therefore $I(C;Y)$ only depends on $p(c|x)$. Gondek et al. [22] extended the IB method for the non-redundant clustering problem by adding the conditional variable Z where Z contains the negative or irrelevant side information. The idea of the conditional IB algorithm (CIB) proposed by Gondek et al. [22] is to maximize $I(C;Y|Z)$, the information that C conveys about Y when given the side information Z. Intuitively, if two clustering C_1 and C_2 convey the same information about Y, i.e. $I(C_1;Y) = I(C_2;Y)$ and the clustering C_1 shares more information with Z than C_2, i.e. $I(C_1;Z) \geq I(C_2;Z)$ then $I(C_1;Y|Z) \leq I(C_2;Y|Z)$ because knowing Z gives more information to C_2 then C_1 to predict Y, i.e. CIB prefers the clustering which is more different from Z. That is the reason why Z is considered as negative side information. Finally, CIB is formulated as follows:

$$\max_{p(c|x)} I(C;Y|Z) \tag{50}$$

$$\text{s.t. } I(C;X) \leq R, \tag{51}$$

$$\forall x. \sum_c p(c|x) = 1, \text{ and } \forall x, c.\ p(c|x) \leq 0 \tag{52}$$

However, like the local minima problem of the *CondEns* algorithm (Algorithm 15), Gondek et al. [23] stated that given Z, there exists a set of optimal clustering solutions for Equ. 50 with different quality, i.e. the preserved information $I(C;Y)$ of some clustering solutions can be very small. Gondek et al. corrected this problem by introducing the constraint $I(C;Y) \geq I_{min}$ [23] to ensure the performance of the clustering solutions. And the new algorithm is named as Coordinated Conditional Information Bottlenack (CCIB), and formulated as follows:

$$\max_{p(c|x)} I(C;Y|Z) \tag{53}$$

$$\text{s.t. } I(C;X) \leq R, \tag{54}$$

$$I(C;Y) \geq I_{min}, \tag{55}$$

$$\forall x. \sum_c p(c|x) = 1, \text{ and } \forall x, c.\ p(c|x) \leq 0 \tag{56}$$

The experimental results in [24] show that the performance of *CondEns* is competitive to the performance of *CCIB*. However, the running time of *CondEns* is significantly smaller than *CCIB*. This is not surprising because the *CondEns* first splits the entire dataset into pre-image sets and then performs clustering on these pre-image sets independently whereas the *CCIB* algorithm searching for the best clustering solution over the whole dataset. Also, the extension and combination steps of *CondEns* are relatively cheap, therefore the *CondEns* complexity is supposed to be less than the *CCIB* complexity.

4 Interactive Scheme

The first passive SSC algorithm with user feedback is proposed by Cohn et al. in [12, 11]. In this approach, the user iteratively provides feedback to a clustering algorithm. The feedback is collected in the form of cannot-link/must-link constraints which the clustering algorithm tries to satisfy in the next iterations by adjusting the distance metric. In detail, the naive Bayes model is used to model the document generation and the parameters of the model are estimated by the EM algorithm. Each document x_i of the dataset $X = \{x_i\}_{i=1}^{N}$ is assumed to be a bag of words w_t and generated from one of the conditional cluster distributions $p(x|c_1), p(x|c_2), ..., p(x|c_K)$ of K clusters $c_1, c_2, ..., c_K$. Let V be the vocabulary set and $N(w_t, x_i)$ be the number of times the word w_t occurs in the document x_i. With the assumption that all the words are independent from each other given the cluster label, the probability $p(x_i)$ of a document x_i is calculated as:

$$p(x_i) = \sum_{j}^{K} p(c_j)p(x_i|c_j) = \sum_{j}^{K} p(c_j) \prod_{w_t \in V} p(w_t|c_j)^{N(x_i,w_t)} \tag{57}$$

The parameters of the model are $\theta = \{p(c_j)\}_j^K \cup \{p(w_t|c_j)\}_{w_t \in V, j=1,...,K}$ which will be estimated by the EM algorithm. Given the model parameter θ and the probability $p(x_i)$ of the document x_i, the probabilistic cluster membership of x_i is estimated by the Bayes' rule:

$$p(c_j|x_i) = \frac{p(x_i|c_j)p(c_j)}{p(x_i)} \tag{58}$$

The EM algorithm for estimating the model parameter θ is given in Algorithm 16. Not that the word probabilities $p(w_t|c_j)$ in the M-step are smoothed with a Laplace prior (a word w_t is assumed to appear in each class c_j at least one) to avoid zero probabilities. Until now, the constraints have not been integrated yet. And the way the authors propose to incorporate the constraints is by adjusting the distance metric. The distance metric is adjusted such that the distance of cannot-link pairs is large enough to classify them into different clusters and vice versa for the must-link pairs. The distance between two documents x_1, x_2 is measured as the probability that these two documents are generated from the same multinomial and this proportional to the Kullback-Leibler divergence to the mean of their multinomial distributions [38]:

$$D_M(x_1||x_2) = |x_1|D_{KL}(\theta_{x_1}, \theta_{x_1,x_2}) + |x_2|D_{KL}(\theta_{x_2}, \theta_{x_1,x_2}) \tag{64}$$

Algorithm 16: The EM algorithm to estimate the naive Bayes model parameter.

Input : Dataset X, number of clusters K.
Output: The parameter model θ which maximize the log-likelihood of the input dataset.

begin

E-step: Fix θ. For all $x_i \in X, c_j \in C$, compute:

$$p(x_i|c_j) = \prod_{w_t \in V} p(w_t|c_j)^{N(x_i,w_t)} \tag{59}$$

$$p(x_i) = \sum_{j}^{K} p(c_j)p(x_i|c_j) \tag{60}$$

$$p(c_j|x_i) = \frac{p(x_i|c_j)p(c_j)}{p(x_i)} \tag{61}$$

M-step: Fix $p(c_j|x_i)$. Compute θ:

$$p(c_j) = \frac{\sum_{i=1}^{|X|} p(c_j|x_i)}{|X|} \tag{62}$$

$$p(w_t|c_j) = \frac{1 + \sum_{i=1}^{|X|} N(w_t, x_i)}{|V| + \sum_{s=1}^{|V|} \sum_{i=1}^{|X|} N(w_s, x_i)p(c_j|x_i)} \tag{63}$$

end

where $|x_i|$ is the length of document x_i, $D_{KL}(\theta_1, \theta_2)$ is the standard Kullback-Leibler diveregence of θ_1, θ_2, $\theta_{x_1} = p(w_t|x_1), \theta_{x_2} = p(w_t|x_2)$ are the word probabilities given x_1, x_2, respectively, and θ_{x_1,x_2} is the following distribution:

$$p(w_t|\theta_{x_1,x_2}) = (p(w_t|x_1) + p(w_t|x_2))/2 \tag{65}$$

In order to integrate the constraint, the standard KL divergence is parameterized with word weights γ_t as follows:

$$D'_{KL}(\theta_{x_1}, \theta_{x_2}) = \sum_{w_t \in V} \gamma_j p(w_t|x_1) log \frac{p(w_t|x_1)}{p(w_t|x_2)} \tag{66}$$

where γ_t is the importance of word w_t to distinguish x_1 and x_2. Replacing D_{KL} by D'_{KL} into Equ. 64 results that the parameterized distance metric D'_M is a function of word weights w_t. Therefore, if (x_1, x_2) is a cannot-link then the distance $D'_M(x_1, x_2)$ between them can be increased by increasing parameters γ_t following the direction of the gradient which is computed as:

$$\frac{\delta D'_M(x_1, x_2)}{\delta \gamma_t} = |x_1| p(w_t|x_1) log \frac{p(w_t|\theta_{x_1,x_2})}{p(w_t|x_1)} + |x_2| p(w_t|x_2) log \frac{p(w_t|\theta_{x_1,x_2})}{p(w_t|x_2)} \tag{67}$$

These word weights are then injected into the E-step of Algorithm 16 by replacing the document probabilities $p(x_i|c_j)$ in Equ. 59 by:

$$p'(x_i|c_j) = \prod_{w_t \in V} p(w_t|c_j)^{\gamma_t N(x_i, w_t)} \tag{68}$$

Other kinds of constraints like must-link constraints are integrated in the similar way. The experimental results show that the performance of the SSC EM algorithm is improved significantly compared to the performance of the unsupervised clustering algorithm with only few constraints, although in principle the EM algorithm can get stuck in some local minima. Besides, the main drawback of the SSC EM algorithm is that the algorithm *passively* receives the constraints from users (simulated by picking a random constraint each time from a set of possible constraints) and in order to achieve a good performance, it requires the user knows what are the most informative constraints to provide to the algorithm and this is not feasible in practice because the user cannot browse thousands (or millions) of constraints to select the best ones.

Instead of passively receiving the feedback from users, the active scheme *Explore-Consolidate* in [5] tries to ask the user through queries to obtain the most informative constraints within the limited number of queries. These informative constraints are used not only in the body of a SSC algorithm to improve the clustering performance but also used to get good estimates of the cluster centroids in the initialization phase of KMEANS based algorithms. Each query is given as a pair of two data objects and then the user replies whether these two data objects must or cannot belong to the same cluster. The active learning scheme is divided into two phases: *Explore* and *Consolidate*. The Explore phase tries to get K pairwise disjoint non-null neighbourhoods as fast as possible where K is the number of clusters. A neighbourhood is defined as a set of data objects belonging to the same cluster. The first data object of the first neighbourhood is picked randomly from the dataset. From that on, a data object x in the remaining data objects which is *farthest* from the existing neighbourhoods is chosen. Then, queries are given by pairs of x with a random data object in each neighbourhood. If x must link (replied by the user) to one of neighbourhoods, assign x to that neighbourhood. Otherwise, create a new neighbourhood with the first member of x. This process is repeated until the number of queries is used up or the number of neighbourhoods equals to K. After the Explore phase, if the query is still allowed, the Consolidate phase is used to identify the neighbourhood (cluster) for the remaining data objects. In this phase, a random object x is picked and then the distances between x and the neighbourhood centers are calculated and sorted in ascending order. Queries are posed as a pair of x and a data object in each neighbourhood in the sorted order of distances between neighbourhoods and x. When a must-link answer is obtained, the algorithm continues with another data object until the number of queries reaches the limit. The maximum number of queries needed to identify the neighbourhood for each object is $K - 1$. The experimental results in the paper shows that integrating the active learning scheme into the constrained clustering algorithms has improved significantly the performance of the constrained clustering algorithms.

A drawback of the *Explore-Consolidate* scheme is that the *Consolidate* phase only selects random points different from the points selected in the *Explore* phase (denoted as the *skeletal* points) to form queries. Mallapragada et al. [34] suggested that selecting the most uncertain points in the *Consolidate* phase to form the queries could improve the performance. Denoting the set of skeletal points as X_s, the certainty $\zeta(x_i, X_s)$ of a point x_i and the skeleton set $X_s \subseteq X$ is defined as the maximum similarity $sim(x_i, x_j)$ between x_i and all other points $x_j \in X_s$:

$$\zeta(x_i, X_s) = \max_{x_j \in X_s} sim(x_i, x_j) \tag{69}$$

Therefore, in each iteration of the *Consolidate* phase, the point x^* with the minimum certainty to the skeleton set is selected:

$$x^* = \operatorname*{argmin}_{x_i \in X \setminus X_s} \zeta(x_i, X_s) \tag{70}$$

A query will be formed by the pair (x^*, x_j) where $x_j \in X_s$ is the nearest point to x^*. x^* is then added to X_s for the next iteration. This process is repeated while the query is allowed. The new scheme *Explore-MinMax* is shown to outperform the *Explore-Consolidate* scheme through experiments on real datasets. Although the *Explore-MinMax* scheme has already searched for the representatives of clusters, they are not guaranteed to be the centroids. Therefore, the new constraint "x_i in a dense region" is added to Equ. 70 to force the selection process favours the centroids because often the centroids are in dense regions [49]:

$$x^* = \operatorname*{argmin}_{x_i \in X \setminus X_s, x_i \text{ in a dense region}} \zeta(x_i, X_s) \tag{71}$$

where a point is in a dense region if it is surrounded by an average number of points which is greater than some threshold. The authors have also modified *Explore-MinMax* for the seed selection problem (the queries are the questions about the labels of queried points, not pair constraints), hence there are no comparison between this scheme and *Explore-MinMax*. However, compared to the random selection process, this scheme is shown to select a larger number of different labels and reduce the number of iterations for KMEANS to converge when KMEANS uses the side information in the form of labelled data extracted by this scheme to initialize the centroids.

Another drawback of the *Explore-Consolidate* scheme is the queries do not take into account the intermediate clustering results (which can be very useful in determining the informative constraints) because this scheme only acquires the informative constraints which are used later by a SSC algorithm. Also, the worst case of this scheme is the situation when in the Explore phase, most constraints obtained are must-link constraints, while in the Consolidate phase, most constraints are cannot-link constraints and this easily happens when the number of clusters is high. An algorithm, named *IG-KMEANS*, solving these problems by considering the intermediate clustering results to select the most informative pairs is then proposed by Huang et al. [29]. *IG-KMEANS* is an extension of *KMEANS* for constraints and optimizes the following objective function:

$$O_{IG} = \rho \sum_{k=1}^{K} \sum_{x_i \in C_k} \psi(x_i, \mu_k) + \tag{72}$$
$$(1-\rho)\Big(\sum_{(x_i,x_j) \in C_=} \psi(x_i, x_j) \mathbf{1}(y_i \neq y_j) +$$
$$\sum_{(x_i,x_j) \in C_{\neq}} (1 - \psi(x_i, x_j)) \mathbf{1}(y_i = y_j) \Big)$$
$$\psi(x_i, x_j) = 1 - \frac{x_i x_j}{||x_i|| ||x_j||} \tag{73}$$

where ρ is the parameter controlling the trade-off between the clustering quality expressed in the first term and the penalty of constraint violation expressed in the second term, and $\psi(x_i, x_j)$ is the distance function between two points x_i, x_j. As for the active learning, given the maximum number of queries Q, and the number of L iterations (predetermined by users), in each iteration, the algorithm selects the best $P = Q/L$ document pairs to form queries based on a gain function measuring how much information obtained when revealing the judgements of the selected document pairs. The constraints formed from these queries will be used in the objective function O_{IG} of the extended *KMEANS* algorithm. Formally, the problem is represented as an optimization problem:

$$\Omega^* = \underset{\Omega}{\operatorname{argmax}} \Lambda(\Omega, \Theta, \Phi) \tag{74}$$
$$\text{s.t. } |\Omega| = P \tag{75}$$

where Ω is a set of document pairs, $|\Omega|$ is the size of Ω, Θ is the current clustering assignments, Φ is the current set of constraints, and Λ is the gain function measuring how much information obtained when knowing the judgements of the document-pair set Ω. Given a document pair $w_i(x_i^1, x_i^2)$ of two documents x_i^1, x_i^2, the judgement of the user will be a value in $J = \{j_m, j_c\}$ where j_m means two documents have a must-link constraint and j_c means two documents have a cannot-link constraint. The gain function can be written as:

$$\Lambda(\Omega, \Theta, \Phi) = \sum_{\vec{J} \in J^P} g(\Omega, \Theta, \Phi, \vec{J}) p(\vec{J} | \Omega, \Theta, \Phi) \tag{76}$$

where $\vec{J} = \{j_1, ..., j_P\}$ is a possible judgement set of the document-pair set Ω, j_i is a possible judgement of the i-th document pair $w_i(x_i^1, x_i^2) \in \Omega$, P is the number of document pairs that can be selected from the current clustering assignment, $g(\Omega, \Theta, \Phi, \vec{J})$ is a judgement gain function showing how much information obtained from the judgement $\vec{J}$ of the document pairs, and $p(\vec{J}|\Omega, \Theta, \Phi)$ is the probability that the judgement $\vec{J}$ is assigned to Ω. Unfortunately, searching for the optimal document-pair set Ω^* is too expensive, therefore the authors propose a another less accurate strategy for selecting pairs with lower complexity as follows. S documents are randomly selected, and ranked based on a document-gain function $G(x_i)$ returning the average information of a document x_i. Then the document x_i^1 with the highest value of the document gain $G(x_i^1)$ is paired with another document x_i^2 in the same cluster with x_i^1 to form a pair $w_i(x_i^1, x_i^2)$ to ask for the judgement. The process is repeated until P pairs are selected. If all pairs are assumed to be selected independently, the document gain information $G(x_i)$ is defined as:

$$G(x_i) = G^{IG}(x_i) = \sum_{j_i \in \{j_m, j_c\}} g^{IG}(w_i(x_i, \mu_i), j_i) p(j_i|w_i(x_i, \mu_i)) \quad (77)$$

$$\sum_{j_i \in J} -log(p(j_i|w_i(x_i, \mu_i)))p(j_i|w_i(x_i, \mu_i)) \quad (78)$$

$$p(j_m|w_i(x_i, \mu_i)) = \frac{s(x_i, \mu_i)}{s(x_i, \mu_i) + s(x_i, \mu_i')} \quad (79)$$

$$p(j_c|w_i(x_i, \mu_i)) = 1 - p(j_m|w_i(x_i, \mu_i)) \quad (80)$$

$$s(x_i, x_j) = \frac{\vec{x}_i \vec{x_j}}{||\vec{x_i}||||\vec{x_j}||} \quad (81)$$

where μ_i is the centroid of the cluster x_i belongs to, $\vec{x_i}$ is the vector representation of x_i, $||x_i||$ is the L_2 norm of x_i, $w_i(x_i, \mu_i)$ is a pair form by x_i and μ_i, and μ_i' is the next nearest centroid to x_i. The independent document-gain G^{IG} can be interpreted as the entropy of the judgement conditioned on the pair $w_i(x_i, \mu_i)$, therefore the pair of the document x_i with high value of $G^{IG}(x_i)$ and another document in the same cluster is supposed to give more information when revealing its judgement. Besides, the probability that x_i belongs to the cluster with the centroid μ_i measured as $p(j_m|w_i(x_i, \mu_i))$ will be high if it is very close to its cluster centroid but far from the next nearest centroid, and low if it is also very close the next nearest centroid. If the independence assumption of pair selection is not given, the authors define the document gain function as:

$$G(x_i) = G^{DG}(x_i) = \sum_{j_i \in \{j_m, j_c\}} g^{DG}(w_i(x_i, \mu_i), j_i) p(j_i | w_i(x_i, \mu_i)) \tag{82}$$

$$\sum_{j_i \in J} -log(p(j_i | w_i(x_i, \mu_i), \Phi)) p(j_i | w_i(x_i, \mu_i), \Phi) \tag{83}$$

$$p(j_m | w_i(x_i, \mu_i), \Phi) = \frac{s_d(x_i, \mu_i | \Phi)}{s_d(x_i, \mu_i | \Phi) + s_d(x_i, \mu_i' | \Phi)} \tag{84}$$

$$p(j_c | w_i(x_i, \mu_i), \Phi) = 1 - p(j_m | w_i(x_i, \mu_i), \Phi) \tag{85}$$

$$s_d(x_i, \mu_i | \Phi) = (1 - \epsilon \frac{Q}{|X|}) s(x_i, \mu_i) + \epsilon \frac{Q}{|X|} \max_{x_j \in \hat{c}_i} s(x_i, x_j) \tag{86}$$

$$\hat{c}_i = \{x_i : x_i \in c_i, \exists x_j . w_i(x_i, x_j) \in \Phi, p(j_m | w_j(x_j, \mu_i) \text{ is high}\} \tag{87}$$

where c_i is the cluster where x_i belongs to, $\hat{c}_i$ is the previously selected documents with high probability to belong to c_i, ϵ is a trade-off constant. The main difference between G^{IG} and G^{DG} is the difference between two similarity functions s and s_d. $s_d(x_i, \mu_i|\Phi)$ is called the dependent cosine similarity, and used to measure the similarity between x_i and μ_i given the set of previously selected documents with high probability to belong to the cluster c_i. s_d is mainly different from s by the second term which computes the maximum similarity of x_i with another previously selected document $x_j \in c_i$. The idea is that if in previously selected documents, there is a document x_j in the same cluster with x_i and x_j is very close to x_i and then the judgements of x_j can provide useful information for the judgements of x_i. And if the probability that $w_j(x_j, \mu_i)$ is a must-link pair is high then the probability that $w_i(x_i, \mu_i)$ is a must-link pair is also high. The ratio $\frac{Q}{|X|}$ is used to emphasize the contribution of each term in the equation of s_d. When Q (the maximum number of queries) is small compared with $|X|$ (the number of documents), then there is only a small number of previously selected documents in c_i. Therefore, the contribution of the similarity between the document x_i and its neighbour x_j should be smaller than the contribution of the similarity between that document and its centroid. And vice versa for the case when Q is large compared to $|X|$. Finally, ϵ is used to control the trade-off between two contributions. The independent/dependent versions of *IG-KMEANS* will be referred as *IIG-KMEANS*, and *DIG-KMEANS*, for short. The experimental results show that the *IG-KMEANS* performance outperforms significantly the *PCKMEANS* algorithm (a KMEANS-based algorithm, represented in Algorithm 8) using the *Explore-Consolidate* scheme proposed by Basu et al. [5] (this algorithm will be referred as *Active-PCKMEANS* for short).

Also, as expected the algorithm with the dependent document-gain function G^{DG} has better performance than algorithm with the independent document-gain function G^{IG}. However, when comparing with *PCKMEANS*, in their KMEAN-based algorithm, the authors do not use the same objective function as in the Basu's KMEAN-based algorithm, therefore it is clear whether the improvement comes from the active learning part or the objective function. Also, the run-time of the information-gain based algorithm is much larger than the *Active-PCKMEANS* algorithm because of the high complexity of computing the membership probabilities $p(j_m|w_i(x_i, \mu_i), \Phi)$.

KMEANS-based algorithms often are not suitable for discovering the non-convex shape clusters, therefore Zhao et al. [54] have proposed a *Constrained-DBSCAN* algorithm which is extension of *DBSCAN* [19] (a density-based clustering algorithm) for the semi-supervised clustering problem. *DBSCAN* requires two parameters: *eps* the radius of neighbourhoods, *minPts* the minimum number points needed to form a cluster. *DBSCAN* first picks randomly a point which has not been visited yet. If the number of points in its *eps*-neighbourhood (the set of points which have a distance to that point less than or equal to *eps*) is less than *minPts*, then that point will be labelled as noise (although it still can be assigned to other cluster later). If this is not the case, a cluster is formed with the initial points are the points in that *eps*-neighbourhood. Next, for each point in that cluster, if its *eps*-neighbourhood size greater than or equal to *minPts* then that neighbourhood is added to the cluster and that point is marked as visited. This process is repeated until the cluster is completely discovered. The next cluster or another noise point will be discovered by continuing the above procedure with a new unvisited point. The *Constrained-DBSCAN* follows the same idea of *DBSCAN* with the extension that adding points to a cluster must guarantee the constraints. The pseudo-code of *Constrained-DBSCAN* is represented in Algorithm 17. First, the algorithm compute the transitive closures of must-link constraints by applying the transitivity property of must-link constraints and update the set of cannot-link constraints by the entailment property of cannot-link constraints and must-link constraints. Then, the algorithm goes through unclassified points and extends (if possible) the cluster of each point but still maintains the constraints in the procedure *Extend_Cluster()* (Algorithm 18). For each starting point x_i of the cluster, all other points which have a must-link constraint with x_i will be added to cluster and form the set of seeds. Recursively, for each other point (or seed) s in the seed set, the points in the transitive closure containing s and in the *eps*-neighbourhood of s (only considered if the neighbourhood size is greater than or equal to *minPts*) are added to the cluster (and the seed set) if the constraints are not violated. Like the *Explore-Consolidate* algorithm of Basu et al. [5], the active selecting procedure for

Constrained-DBSCAN, named *Active-Selecting-DBSCAN*, is not integrated into *Constrained-DBSCAN* but only used to obtain the set of informative constraints and then passes them to *Constrained-DBSCAN*. The authors define two types of data points for the purpose of selecting constraints as follows. If the size of the *eps*-neighbourhood of a point is greater than or equal to $minPts$, then that point is a core point, otherwise it is a border point. The main idea of *Active-Selecting-DBSCAN* is to try to obtain the constraints which can help to determine the boundaries of clusters and identify at least one core point for each cluster. *Active-Selecting-DBSCAN* first tries to build the core and border point sets. Then it will identify the constraints on the pairs of a core point and a border point for determining the cluster boundaries and the constraints on the pairs of two farthest core points for having at least one core point for each cluster as fast as possible. The *Active-Selecting-DBSCAN* is represented in Algorithm 19. The experimental

---

Algorithm 17: Constrained-DBSCAN

Input : Dataset X, the set of must-link constraints $C_=$, the set of cannot-link constraints $C_{\neq}$, the radius *eps*, the minimum number of points in a neighbourhood $minPts$.

Output: A set of clusters and a set of noise points.

```
begin
    1. Initialize all objects in X as UNCLASSIFIED.
    2. Preprocess the constrains: compute the set of transitive closures
    TCS = {TCS_i} of must-link constraints, and update the set of
    cannot-link constraints C_≠.
    3. clusterId := 0.
    4. Discover clusters:
    for each point x_i ∈ X do
        if y_i is UNCLASSIFIED then
            // Expand the cluster
            Compute x_i's eps-neighbourhood N_i.
            if size of N_i < minPts then
                y_i = NOISE
                Continue the loop with the next x_i.
            end
            Expand_Cluster(X, x_i, eps, minPts, TCS, C_≠, clusterId).
            clusterId = clusterId + 1.
        end
    end
end
```

Algorithm 18: Expand-Cluster

Input : Dataset X, starting point x_i, transitive closures of must-link constraints TCS, set of cannot-link constraints $C_{\neq}$, radius eps, minimum number of points in a neighbourhood $minPts$, current cluster id $clusterId$.

Output: The extended cluster.

```
begin
    seeds = ∅.
    if ∃CS_j ∈ TCS. x_i ∈ TCS_j then
        for each point x_t ∈ TCS_j do
            y_t = clusterId
            seeds = seeds ∪ {x_t}
        end
    else
        y_i = clusterId
        seeds = seeds ∪ x_i
    end
    while seeds ≠ ∅ do
        Get the first object s ∈ seeds.
        if ∃TCS_j ∈ TCS. x_i ∈ TCS_j then
            for each point x_t ∈ TCS_j do
                if y_t is NOISE or UNCLASSIFIED then
                    y_t = clusterId
                    seeds = seeds ∪ {x_t}
                end
            end
        end
        Compute s's eps-neighbourhood N_s.
        if size of N_s ≥ minPts then
            for each object x_t ∈ N_s do
                if adding x_i into seeds does not violate cannot-link
                constraints, and y_i is NOISE or UNCLASSIFIED
                then
                    x_t = clusterId
                    seeds = seeds ∪ {x_t}
                end
            end
        end
        seeds = seeds \ {s}
    end
end
```

Algorithm 19: Active-Selecting-DBSCAN

Input : Dataset X, radius *eps*, minimum number of points in a neighbourhood $minPts$, maximum number of queries Q

Output: A set of instance-level constraints.

```
begin
    1. Compute the core point set CS and the border point set BS in
    X with respect to eps and minPts.
    2. consSet = ∅. // The constraint set
    3. SCS = ∅. // The selected core point set.
    4. Obtain the constraints:
    while queries are allowed do
        // obtain cannot-link constraints.
        if SCS = ∅ then
            Pick the first core point x_i in CS randomly, and add x_i into
            SCS.
        else
            Pick the point x_i farthest from SCS.
            for each point x_j ∈ SCS do
                Ask for the judgement (must-link or cannot-link) of the
                pair (x_i, x_j).
                Add the constraint (x_i, x_j) to consSet.
            end
            SCS = SCS ∪ {x_i}.
        end
        // obtain the must-link constraints.
        Pick the point x_1 in BS which is nearest from x; ask for the
        judgement of the pair (x_i, x_1); add the constraint (x_i, x_1) to
        consSet.
        Pick the point x_2 in BS which is nearest from x; ask for the
        judgement of the pair (x_i, x_2); add the constraint (x_i, x_2) to
        consSet.
    end
    return conSet
end
```

results show that *Active-Constrained-DBSCAN* (*Constrained-DBSCAN* + *Active-Selecting-DBSCAN*) outperforms significantly *Constrained-DBSCAN* in all cases. Comparing with *Active-PCKMEANS* [5], *Active-IG-KMEANS* [29], *Active-Constrained-DBSCAN* is much better than *Active-PCKMEANS*, and slightly better than *Active-IG-KMEANS*. The reason is that *Active-PCKMEANS* easily uses up the number of queries if a lot of cannot-link constraints are obtained and this happens when the number of clusters is large. In order to compare the active selecting strategies, the authors implement *Constrained-DBSCAN* with *Explore-Consolidate* scheme [5], named *EC-Constrained-DBSCAN* and *PCKMEANS* with *Active-Selecting-DBSCAN*, named *Active-D-PCKMEANS*. And the result is that *Active-Constrained-DBSCAN* is much better than *EC-Constrained-DBSCAN*, *Active-D-PCKMEANS* is also significantly better than *Active-PCKMEANS*. The reason can come from the fact that the constraints obtained by *Active-Selecting-DBSCAN* containing good information about the cluster boundaries while it is not the case of *Explore-Consolidate*. Also, *Active-D-PCKMEANS* is slightly worse than *Active-Constrained-DBSCAN* because *DBSCAN* is more suitable for the case of overlapping clusters than *KMEANS*. Finally, the run-time of *Active-Constrained-DBSCAN* (linearly with the number of constraints) is substantially smaller than the run-time of *Active-IG-KMEANS* due to the fact that the constraints are obtained for *Active-Constrained-DBSCAN* only once before executing the semi-supervised clustering process while in *Active-IG-KMEANS*, the selecting constraint process and the semi-supervised clustering process are iterated alternatively.

An ensemble-based selection procedure for identifying the most informative constraints is proposed by Greene et al. [26]. The ensemble-based procedure is split into two phases: imputing constraints from pairwise co-associations and selecting informative constraints. First, the co-association matrix A is built by τ clustering results of a base clustering algorithm on τ samples of the input dataset X (without replacement) as in Algorithm 20. The value of a matrix cell A_{ij} denotes the fraction of base clusterings C_t in which two points x_i and x_j are assigned to the same cluster. For a sufficient large number of base clusterings, $A_{ij} \approx 1$ indicates that x_i and x_j should belong to the same cluster, while $A_{ij} \approx 0$ implies that x_i and x_j should be in different clusters. When $A_{ij} \approx 0.5$, the relationship between x_i and x_j is highly uncertain and this can happen when they are both at the boundaries. Given two thresholds κ_m and κ_c of A_{ij} where κ_m/κ_c is the minimum/maximum confidence that two points are in the same cluster to form a must-link/cannot-link constraint, the sets of imputed must-link constraints

$C'_=$ and of cannot-link constraints $C'_{\neq}$ are obtained as follows:

$$C'_=\{(x_i, x_j)|A_{ij} \geq \kappa_m\} \tag{88}$$
$$C'_{\neq}\{(x_i, x_j)|A_{ij} \leq \kappa_c\} \tag{89}$$

Next, $C_=$ is updated as its transitive closure by applying the transitivity

Algorithm 20: Build the co-association matrix $\boldsymbol{A}$.

Input : Input dataset $X = \{x_i\}_{i=1}^N$, number of clusters K
Output: The co-association matrix $\boldsymbol{A}$.
begin
  1. Initialize a zero $N \times N$ matrix $\boldsymbol{A}$. 2.
  for $t = 1$ *to* τ **do**
    2.1 - Draw a sample of points X_t by random sampling without replacement.
    2.2 - Generate a base clustering C_t by clustering the sample X_t.
    2.3 - Classify the remaining points $x_i \in X \setminus X_t$ based on the clusters in C_t.
    2.4 - For each pair (xi, xj) assigned to the same cluster in C_t, update $\boldsymbol{A}$: $A_{ij} = A_{ij} + 1/\tau$.
  end
end

of must-link constraints $((x_i, x_j)$ and (x_j, x_t) in $C_= \Rightarrow (x_i, x_t) \in C_=)$. Then, the K neighbourhoods and their representatives are computed from $C_=$. The first representative is chosen as the median of the largest neighbourhood in $C_=$. Each of the remaining $K-1$ representatives is selected as the median of the largest remaining neighbourhood with the condition that a cannot-link constraint exists between that median and the previously selected representatives. This step results in an initial clustering $C_0 = \{C_1^0, C_2^0, ..., C_K^0\}$ where C_i^0 is the i-th cluster (or neighbourhood) of the clustering C_0. The pseudocode of this step is illustrated in Algorithm 21. In the selection phase, the set of constraints will be expanded by selecting the most uncertain point to form the queries. For each point $x_i \in X$ and a cluster $C_0^t \in C_0$, the association S_{tc} of x_i and C_0^t is calculated as the average of the co-association of x_i and all member $x_j \in C_0^t$:

$$S_{it} = \frac{1}{|C_0^t|} \sum_{x_j \in C_0^t} A_{ij} \tag{90}$$

Algorithm 21: Constraint set initialization phase.

Input : Imputed constraint sets $C_=$ and $C_{\neq}$.
Output: An initial clustering $C_0 = \{C_0^1, C_0^2, ..., C_0^K\}$.
begin
1. Update $C_=$ as its transitive closure and compute the neighbourhoods from the updated $C_=$.
2. Choose the first representative r_1 to be the median of the largest neighbourhood.
3.
for $t = 2$ *to* K **do**
Select r_t as the median of the next largest neighbourhood with the condition that a cannot-link constraint exists between r_t and each of $\{r_1, ..., r_{t-1}\}$.
end
return *the clustering* $C_0 = \{C_0^1, C_0^2, ..., C_0^K\}$ *where* $r_t \in C_0^t$ *together with any other object with a must-link constraint to* r_t.
end

and the certainty $w(x_i)$ of assigning x_i to a cluster in C_0 is measured as the margin between the cluster C_0^a with the highest association with x_i and the cluster C_0^b with the second highest association and formally defined as:

$$w(x_i) = \frac{2S_{ia}}{S_{ia} + S_{ib}} - 1 \tag{91}$$

However, if the uncertain points are selected based on $w(x_i)$, it can result in the situation that a lot of constraints for *only* a specific class are generated. And this can lead to poor performance when the number of queries allowed is small. The authors solve this problem by considering also cluster sizes as weights for co-association values, and define the new certainty criterion as:

$$w'(x_i) = \frac{2T_{ia}}{T_{ia} + T_{ib}} - 1 \tag{92}$$

$$T_{it} = \frac{|C_0^t|}{\sum_j |C_0^j|} S_{it} \tag{93}$$

where T_{ia}, T_{ib} are the highest and the second highest weighted object-cluster association values of x_i. The weights based on cluster sizes are added to prioritize the objects that will be assigned to small clusters, therefore it can reduce the effect of selecting a lot of points belonging to the same cluster with a large size. Then, when the most uncertain point x_i (the point with the minimum value of $w'(x_i)$) is selected, its correct cluster will be discovered by posing the queries about the relation of x_i and the K representatives. At most $K-1$ queries are needed to identify the cluster of x_i because when $K-1$ cannot-link constraints between x_i and $K-1$ representatives are obtained, it can be inferred that x_i belong to the remaining cluster. The selection procedure is illustrated as in Algorithm 22. With a large enough number

---

Algorithm 22: Constraint set expansion phase.

Input : cluster-object co-association matrix $\boldsymbol{S}$.
Output: two sets of new constraints $C'_=$ and $C'_{\neq}$.
begin
  $C'_= = \emptyset, C'_{\neq} = \emptyset$.
  while *queries are allowed* **do**
    1. Select the most uncertain object x_i with minimum value of $w'(x_i)$, calculated as in Equ. 92.
    2.
    $y_i = UNCLASSIFIED.$
    for *each cluster C_0^t in descending order of S_{it}* **do**
      Ask the judgement for the constraint (x_i, r_t).
      if *(x_i, r_t) is a must-link constraint* **then**
        $y_i = t.$
        Break the for loop.
      else
        $C'_{\neq} = C'_{\neq} \cup \{(x_i, \mu_t)\}$
      end
      if *the number of queries are used up* **then**
        return $C'_=, C'_{\neq}$.
      end
    end
    if $y_i = UNCLASSIFIED$ **then**
      $y_i =$ the label of the remaining cluster.
    end
    Assign x_i the cluster with the label of y_i.
    $C'_= = C'_= \cup \{(x_i, \mu_t)\}$.
  end
  return $C'_=, C'_{\neq}$.
end

---

of ensembles, the experiments on real datasets show that most of imputed constraints are correct (greater than 90% in most cases) according to the true labels of object. Also, in some datasets, the percentage of imputed constraints obtained on the total number of constraints is relatively large, e.g. 38% of the total number of constraints. Finally, compared with the *Explore-Consolidate* scheme [5], this ensemble-based approach outperforms in all datasets, and this scheme is especially better than the *Explore-Consolidate* scheme on the tests with small numbers of queries.

Until now, it seems that adding constraints to a basic clustering algorithm always improves performance. Unfortunately, this intuition is not always correct and proven through experiments in [26, 15]. Davidson et al. [15] explain the adverse effect of noiseless constraints through two measures of constraint set utility: *informativeness* and *coherence*. The *informativeness* is the amount of information given by the constraint set and cannot be determined by the algorithm by itself, and the *coherence* is the amount of the agreement between the constraints themselves according to a given metric. Let P^* be the partition (or clustering) that globally minimizes the objective function of a clustering algorithm $\mathcal{A}$ with no constraints. And C^* is a constraint set of $\binom{n}{2}$ must-link and cannot-link constraints that completely specifies P^*. The idealized informativeness of a given constraint set C is the fraction of constraints in C that are violated by C^*. The idea is that if the constraint set C is noiseless, then any constraint $c \in C$ which is not satisfied by the best partition P^* (with no constraint) will give new information to the algorithm $\mathcal{A}$. However, in practice, P^* is unknown, therefore a local optimum partition $P_{\mathcal{A}}$ of the algorithm $\mathcal{A}$ is used instead. This leads to definition of the approximate informativeness as follows:

$$I_{\mathcal{A}}(C) = \frac{1}{|C|} \sum_{c \in C} unsat(c, P_{\mathcal{A}}) \tag{94}$$

where $unsat(c, P_{\mathcal{A}})$ is 1 if the constraint c is satisfied by P, and 0 otherwise. In contrast to *informativeness*, *coherence* is independently defined from the algorithm $\mathcal{A}$ but it is dependent on a metric $\mathcal{D}$. And the definition of *coherence* is originated from the following view. A must-link constraint $c_=(x_i, x_j)$ or a cannot-link constraint $c_{\neq}(x_i, x_j)$ of x_i and x_j can be considered as an attractive or repulsive force along the line connecting x_i and x_j, respectively. Therefore, if a must-link constraint and a cannot-link constraint have contradictory forces in the same region, they will cause a conflict. In other words, if their forces are mostly overlapped each other, they are highly incoherent. Consider two vectors $\vec{a}(a_1, a_2)$ and $\vec{b}(b_1, b_2)$, the projection $\vec{p}(p_1, p_2)$ of $\vec{a}$ on $\vec{b}$ is computed as:

$$\vec{p} = |\vec{a}| cos(\theta) \frac{\vec{b}}{|\vec{b}|} \tag{95}$$

where θ is the angel between the two vectors. And the overlap $overlap_{\mathcal{D}}^{b}(a)$ of two vectors when projecting $\vec{a}$ on $\vec{b}$ under the metric $\mathcal{D}$ is calculated as:

$$overlap_{\mathcal{D}}^{b}(a) = \begin{cases} 0 & \text{if } \mathcal{D}(b_2, b_1) \leq \mathcal{D}(b_2, p_2),\ \mathcal{D}(b_2, b_1) \leq \mathcal{D}(b_2, p_1) \\ \mathcal{D}(b_1, p_2) & \text{if } \mathcal{D}(b_2, p_2) < \mathcal{D}(b_2, b_1),\ \mathcal{D}(b_2, p_1) \geq \mathcal{D}(b_2, b_1) \\ \mathcal{D}(p_1, p_2) & \text{if } \mathcal{D}(b_2, p_2) < \mathcal{D}(b_2, b_1),\ \mathcal{D}(b_2, p_1) < \mathcal{D}(b_2, b_1) \end{cases} \tag{96}$$

From the above formula, the *coherence* $COH_{\mathcal{D}}(C)$ of a constraint set C is defined as:

$$COH_{\mathcal{D}}(C) = \frac{\sum_{m \in C_=, c \in C_{\neq}} \mathbf{1}[overlap_{\mathcal{D}}^{c}(m) = 0 \text{ and } overlap_{\mathcal{D}}^{m}(c) = 0]}{|C_=||C_{\neq}|} \tag{97}$$

where $\mathbf{1}[true] = 1$ and $\mathbf{1}[false] = 0$. Finally, from the experimental results in the paper, it can be observed that most constraint sets with high *informativeness* and *coherence* improve the clustering performance, whereas the *incoherent* sets with low *informativeness* result in an adverse effect.

5 Open Issues

In 2007, Wagstaff has discussed three main issues of constrained clustering in [50]. They are the questions about how to evaluate the utility of a given constraint set, how to reduce the cost of acquiring the constraints, and how to propagate the constraint information to near regions to avoid collecting redundant constraints. Most of the works in the next subsections of this section are based on the Wagstaff's paper [50].

5.1 How to evaluate the utility of a given constraint set?

Davidson et al. [15] have pointed out that integrating constraints into clustering algorithms does not always help to improve the performance. In some cases, the constraints can even cause the adverse effect. Table 5.1 (extracted from [15]) shows the fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy of four constrained-clustering algorithms: *COP-KMEANS* [51], *PKM* [9], *MKM*[9], *MPKM* [9] on four UCI datasets [10]. It can be seen that in some cases, the fraction of the constraint sets the decrease the performance is extremely high, e.g. 87% in the case of *MKM* on the *Wine* dataset. Two measures *Informativeness* and *Coherence* have been

Data Set	*CKM* [51]	*PKM* [9]	*MKM* [9]	*MPKM* [9]
Glass	28%	1%	11%	0%
Ionosphere	26%	77%	0%	77%
Iris	29%	19%	36%	36%
Wine	38%	34%	87%	74%

Table 1: Fraction of 1000 randomly selected 25-constraint sets that caused a drop in accuracy of four constrained-clustering algorithm.

proposed by Davidson et al. [15] to evaluate the utility of a constraint set. From the experiments, it can be observed that the constraint sets with high *informativeness* and *coherence* often improve the performance. However, it is not always the case that these measures can explain the constrained clustering results on some datasets. Table 2 presents the experimental results on fully coherent constraint sets with different values of *informativeness* (low or high), the constraint sets with high informativeness improve significantly the performance of all algorithms on the *Iris* dataset while on the *Wine* dataset, these constraint sets do not show any effect on the clustering performance [15]. Therefore, in order to evaluate the utility of a constraint set, more

	CKM [51]		*PKM* [9]		*MKM* [9]		*MPKM* [9]	
Data Set	H-Inf.	L-Inf.	H-Inf.	L-Inf.	H-Inf.	L-Inf.	H-Inf.	L-Inf.
Ionosphere	58.9%	58.9%	58.8%	58.7%	58.9%	58.9%	93.9%	93.5%
Iris	89.2%	88.1%	88.1%	86.7%	92.9%	89.2%	93.9%	93.5%

Table 2: Average accuracy of four algorithms on two datasets with low informativeness (L-Inf.) and high informativeness (H-Inf.) constraint sets.

efforts must be spent for identifying other constraint set attributes (like *informativeness* and *coherence*) as well as the procedures for predicting the utility of constraint sets from their attributes.

5.2 How to reduce the cost of acquiring the constraints?

Usually, constraints are provided once by a supervisor before executing the constrained clustering algorithm, or obtained by asking the supervisor interactively. In both cases, the number of constraints are very limited because of the expensive cost for collecting the constraints. Hence, minimizing the number of constraints is one of the most important issues in SSC. The studies in literature have shown that letting the algorithm actively ask the supervisor what it wants to know can reduce the number of constraints much better than requiring the supervisor knows what constraints he/she should supply to the algorithm. The algorithms proposed for solving this problem have been discussed in Section 4 like: using the farthest distance [5], information gain [29], density [54] and co-association confidence [26] to select the most informative constraints. However, until now, there is still no work that integrates different types of constraint utility measures into a constraint selection procedure to select the most informative constraints. Also, in the last 10 years, most works in literature only focus on exploiting the two classic instance-level constraint types: must-link and cannot-link constraints [51]. Thus, more efficient types of high-level constraints which can be equivalent to a batch of instance-level constraints are needed to be studied.

5.3 How to propagate the constraint information?

Some works [31, 52, 9, 2] have been done in literature to propagate the constraints to near regions. The idea is to bring the neighbourhoods of two points in a must-link constraint near to each other. For example, if $(x_i, x_j) \in C_=$ and x_i is very near x_a then when distance metric is learned, the distance between x_i and x_j will be shrunk to bring x_i near to x_j. Because of the constraint propagation, the distance between x_j and x_a will be shrunk too, therefore x_j and x_a are likely assigned in the same cluster and similarly for the case of cannot-link constraints. This approach has been shown to be successful when the constraints and the distance metric are consistent [31, 52, 9, 2]. However, this condition is not always valid. A counterexample is the UCI dataset *tic-tac-toe* [10]. In this dataset, each item is a 3×3 board representing the current state of the game. x, o denote the cells occupied by the first and second player, respectively. The goal is to classify boards into two clusters: the cluster of boards that the first user wins, and the other one consists of boards that the first user loses or draws. As shown in Fig. 7 extracted from [50], although *Board A* and *Board C* are in the same cluster, their distance is very large. In contrast, *Board A* and *Board B* belong to different clusters, but their distance is much smaller. If there are a cannot-link constraint for the pair (*Board A*, *Board B*) and a must-link constraint for the pair (*Board A*, *Board C*) then propagating constraints to near regions in this case is supposed not to improve the performance (or even decrease the performance) because it will bring *Board B* near to *Board C* by propagating the must-link constraint between *Board A* and *Board C*. Then, if not careful, *Board B* is even brought closer to *Board C* than *Board A* when *Board B* is forced to be far from *Board A* but in the direction towards *Board C* by the cannot-link constraint between *Board B* and *Board A*. Therefore, identifying datasets in which propagating constraints is correct and how far

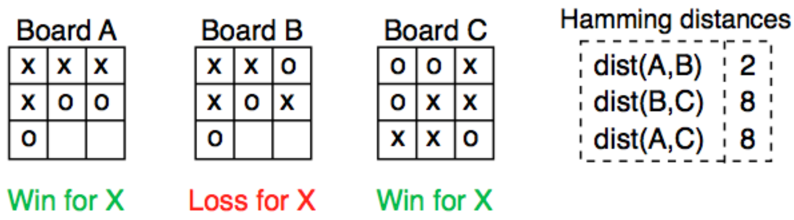


Figure 7: Three boards of the *tic-tac-toe* dataset and their Hamming distances.

the constraints should be propagated are the challenging issues for this field.

References

[1] A. Banerjee and J. Ghosh. Clustering with balancing constraints. *Constrained clustering: advances in algorithms, theory, and applications*, page 171, 2008.

[2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(1):937, 2006.

[3] S. Basu. *Semi-supervised clustering: Probabilistic models, algorithms and experiments*. PhD thesis, THE UNIVERSITY OF TEXAS AT AUSTIN, 2006.

[4] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 19–26, 2002.

[5] S. Basu, A. Banerjee, and R.J. Mooney. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the SIAM international conference on data mining*, pages 333–344, 2004.

[6] S. Basu, M. Bilenko, and R.J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68. ACM, 2004.

[7] S. Basu, I. Davidson, and K.L. Wagstaff. *Constrained clustering: Advances in algorithms, theory, and applications*. Chapman & Hall/CRC, 2008.

[8] D.P. Bertsekas. *Linear network optimization*. MIT Press, 1991.

[9] M. Bilenko, S. Basu, and R.J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 11. ACM, 2004.

[10] C.L. Blake and C.J. Merz. Uci repository of machine learning databases. `http://archive.ics.uci.edu/ml/`.

[11] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, pages 17–31, 2008.

[12] David Cohn, Rich Caruana, and Andrew Mccallum. Semi-supervised clustering with user feedback. Technical report, Cornell University, 2003.

[13] T.M. Cover, J.A. Thomas, and MyiLibrary. *Elements of information theory*, volume 6. Wiley Online Library, 1991.

[14] I. Davidson and SS Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *Proceedings of the Fifth SIAM International Conference on Data Mining*, volume 119, page 138. Society for Industrial Mathematics, 2005.

[15] I. Davidson, K. Wagstaff, and S. Basu. Measuring constraint-set utility for partitional clustering algorithms. *Knowledge Discovery in Databases: PKDD 2006*, pages 115–126, 2006.

[16] Ian Davidson and Sugato Basu. A survey of clustering with instance level constraints. *ACM Transactions on Knowledge Discovery from Data*, 2007.

[17] D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364, 1977.

[18] A. Demiriz, K.P. Bennett, and P.S. Bradley. Using assignment constraints to avoid empty clusters in k-means clustering. *Constrained clustering: advances in algorithms, theory, and applications*, page 201, 2008.

[19] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. Portland: AAAI Press, 1996.

[20] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6:721–741, 1984.

[21] D. Gondek. Non-redundant data clustering. *Constrained clustering: advances in algorithms, theory, and applications*, page 245, 2008.

[22] D. Gondek and T. Hofmann. Conditional information bottleneck clustering. In *3rd ieee international conference on data mining, workshop on clustering large data sets*, pages 36–42. Citeseer, 2003.

[23] D. Gondek and T. Hofmann. Non-redundant data clustering. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on*, pages 75–82. IEEE, 2004.

[24] D. Gondek and T. Hofmann. Non-redundant clustering with conditional ensembles. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 70–77. ACM, 2005.

[25] S. Gordon, H. Greenspan, and J. Goldberger. Applying the information bottleneck principle to unsupervised clustering of discrete and continuous image representations. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 370–377. IEEE, 2003.

[26] D. Greene and P. Cunningham. Constraint selection by committee: An ensemble approach to identifying informative constraints for semi-supervised clustering. *Machine Learning: ECML 2007*, pages 140–151, 2007.

[27] J.M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. 1968.

[28] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 50. ACM, 2004.

[29] R. Huang, W. Lam, and Z. Zhang. Active learning of constraints for semi-supervised text clustering. In *Proceedings of the SIAM International Conference on Machine Learning*, pages 113–124, 2007.

[30] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[31] D. Klein, S.D. Kamvar, and C.D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 307–314. Citeseer, 2002.

[32] Y. Liu, R. Jin, and A.K. Jain. Boostcluster: boosting clustering by pairwise constraints. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 450–459. ACM, 2007.

[33] Z. Lu and T.K. Leen. Pairwise constraints as priors in probabilistic clustering. *Constrained clustering: advances in algorithms, theory, and applications*, page 59, 2008.

[34] P.K. Mallapragada, R. Jin, and A.K. Jain. Active query selection for semi-supervised clustering. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[35] M. Meila. Comparing clusterings by the variation of information. *Learning theory and kernel machines*, pages 173–187, 2003.

[36] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceeding of the 2001 Conference*, pages 849–856, 2001.

[37] D. Pelleg and D. Baras. K-means with large and noisy constraint sets. *Machine Learning: ECML 2007*, pages 674–682, 2007.

[38] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190. Association for Computational Linguistics, 1993.

[39] Nielsen Marketing Research. *Category Management: Positioning Your Organization to Win*. McGraw-Hill, 1993.

[40] Vaithyanathan S. and Gondek D. Clustering with informative priors. Technical report, IBM Almaden Research Center, 2002.

[41] R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of statistics*, pages 1651–1686, 1998.

[42] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[43] C.E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[44] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Gaussian mixture models with equivalence constraints. *Constrained clustering: advances in algorithms, theory, and applications*, page 33, 2008.

[45] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30, 1973.

[46] A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

[47] N. Tishby, F.C. Pereira, and W. Bialek. The information bottleneck method. *Arxiv preprint physics/0004057*, 2000.

[48] E.M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.

[49] V.V. Vu, N. Labroche, and B. Bouchon-Meunier. Active learning for semi-supervised k-means clustering. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE International Conference on*, volume 1, pages 12–15. IEEE.

[50] K. Wagstaff. Value, cost, and sharing: Open issues in constrained clustering. *Knowledge Discovery in Inductive Databases*, pages 1–10, 2007.

[51] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrodl. Constrained k-means clustering with background knowledge. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 577–584. Citeseer, 2001.

[52] E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. *Advances in neural information processing systems*, pages 521–528, 2003.

[53] Y. Yang and B. Padmanabhan. Segmenting customer transactions using a pattern-based clustering approach. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 411–418. IEEE, 2003.

[54] W. Zhao, Q. He, H. Ma, and Z. Shi. Effective semi-supervised document clustering via active learning with instance-level constraints. *Knowledge and Information Systems*, pages 1–19, 2011.