An Efficient Semi-Supervised Clustering Algorithm with Sequential Constraints

Jinfeng Yi[1], Lijun Zhang[2], Tianbao Yang[3], Wei Liu[1], and Jun Wang[4]
[1]IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
[2]National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
[3]Department of Computer Science, University of Iowa, Iowa City, IA 52242, USA
[4]Data Science, Alibaba Group, Seattle, WA 98101, USA
jinfengy@us.ibm.com, zhanglj@lamda.nju.edu.cn, tianbao-yang@uiowa.edu,
weiliu@us.ibm.com, wongjun@gmail.com

ABSTRACT

Semi-supervised clustering leverages side information such as pairwise constraints to guide clustering procedures. Despite promising progress, existing semi-supervised clustering approaches overlook the condition of side information being generated sequentially, which is a natural setting arising in numerous real-world applications such as social network and e-commerce system analysis. Given emerged new constraints, classical semi-supervised clustering algorithms need to re-optimize their objectives over all data samples and constraints in availability, which prevents them from efficiently updating the obtained data partitions. To address this challenge, we propose an efficient dynamic semi-supervised clustering framework that casts the clustering problem into a search problem over a feasible convex set, *i.e.*, a convex hull with its extreme points being an ensemble of m data partitions. According to the principle of ensemble clustering, the optimal partition lies in the convex hull, and can thus be uniquely represented by an m-dimensional probability simplex vector. As such, the dynamic semi-supervised clustering problem is simplified to the problem of updating a probability simplex vector subject to the newly received pairwise constraints. We then develop a computationally efficient updating procedure to update the probability simplex vector in $O(m^2)$ time, irrespective of the data size n. Our empirical studies on several real-world benchmark datasets show that the proposed algorithm outperforms the state-of-the-art semi-supervised clustering algorithms with visible performance gain and significantly reduced running time.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining; I.5.3 [**Pattern Recognition**]: [Clustering]

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

ACM 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783389.

Keywords

Semi-supervised clustering, sequential constraints, convex hull, probability simplex.

1. INTRODUCTION

Although data clustering has been successfully applied to plenty of domains [27, 5, 10, 42, 37], it remains as an ill-posed problem due to its unsupervised nature [31]. Semi-supervised clustering [2] could address this limitation by effectively exploring available supervision to guide the clustering process. Such supervision, also known as side information, is often expressed in the form of pairwise constraints, *i.e.*, *must-links* between the data pairs that belong to the same cluster and *cannot-links* between the data pairs that belong to different clusters. The key idea of semi-supervised clustering algorithms is to find the optimal data partition which is consistent with both the given pairwise constraints and inherent feature representation of data objects to be clustered.

Despite the promising progress, one issue often overlooked by existing semi-supervised clustering approaches is how to efficiently update the clustering results when the pairwise constraints are dynamic, *i.e.*, the new pairwise constraints are generated sequentially. This condition stands natural and is closely related to many real-world applications. For example, one representative application in social network analysis is to identify user communities based on users' profiles as well as their social connections. If we respectively treat user profiles and connections as features and pairwise constraints, this application is essentially a semi-supervised clustering problem. Since new connections are being formed over time, user communities should also be frequently updated. Similar situations also occur in various real-world e-commerce platforms, which typically require to group items or customers based on their profiles (*i.e.*, features) and dynamic co-purchasing histories (*i.e.*, pairwise constraints).

We notice that although the subject of evolving clustering has been intensively studied, to the best of our knowledge, no previous study has focused on the problem of efficiently updating semi-supervised clustering results given sequential constraints. To tackle this challenging problem, in this paper we propose an efficient dynamic semi-supervised clustering framework for large-scale data mining applications [48, 22, 40, 41]. The key idea is to cast the semi-supervised clustering problem into a search problem over a convex hull. More

specifically, the proposed framework consists of two components: (i) an offline step for constructing a convex hull, and (ii) an online step for efficiently updating the clustering results when the new pairwise constraints come in. In the first step, we employ the ensemble clustering technique [51] to generate m ensemble partitions of n input data points to be clustered. According to the principle of ensemble clustering [51, 54, 25], the optimal data partition can be approximated by a convex combination of the ensemble partitions. Since all the convex combinations of m ensemble partitions form a convex hull with m extreme points, the optimal data partition should lie in the inner space spanned by this convex hull. This observation relaxes the data clustering problem to a novel problem of learning m combination weights, or equivalently, an m-dimensional probability simplex vector $\boldsymbol{\gamma}$[1]. Since m is usually significantly smaller than the number of data points n, it enables us to efficiently update the data partitions through updating the m-dimensional probability simplex vector $\boldsymbol{\gamma}$. In the second step of the framework, we design an efficient updating scheme that is able to update $\boldsymbol{\gamma}$ in a time complexity of $O(m^2)$, which does not depend on the number of the data points to be clustered.

Compared with the existing approaches engaged in semi-supervised clustering, our proposed approach owns the following advantages:

1. By applying the ensemble clustering technique, our approach is able to exploit the strengths of different ensemble partitions and meanwhile compensate for their limitations. This can help us to achieve more robust clustering results.

2. By simplifying the problem of clustering n data points to the novel problem of learning an m-dimensional vector $\boldsymbol{\gamma}$, our approach enjoys a low time complexity, which is independent of the total number of data points, in updating $\boldsymbol{\gamma}$. This allows for updating large-scale clustering results in an extremely efficient way.

To evaluate the performance of the proposed dynamic semi-supervised clustering approach, we conduct empirical studies on several real-world benchmark datasets. The experimental results show that the proposed approach surpasses the competing approaches in both accuracy and efficiency.

2. RELATED WORK

In this section, we divide the related work into three categories, namely semi-supervised clustering, clustering according to user's feedback, and dynamic network clustering.

Most semi-supervised clustering methods can be categorized into two main groups [6]: constrained clustering methods and distance metric learning based methods. The constrained clustering methods employ side information to confine the solution space, and only seek feasible data partitions consistent with given constraints. Among them, hard constraints based methods only consider the cluster assignments such that all the constraints are strictly satisfied. For instance, Wagstaff *et al.* [57] modified the K-means clustering and self-organizing map algorithms to adjust the cluster memberships towards the given pairwise constraints. In [50], a generalized Expectation Maximization (EM) algorithm was applied to ensure that only the mixture models matching all the constraints are considered. The hard constraints based methods tend to be more sensitive to noise since some constraints may make the corresponding clustering problems infeasible [18]. To overcome this issue, a lot of studies have treated side information as soft constraints [3, 4, 17, 36, 45]. Instead of satisfying all the constraints, the soft constraints based methods aim to preserve those constraints as many as possible, while penalizing the number of violated constraints. In [3, 45, 4], probabilistic models were proposed to deal with semi-supervised clustering tasks, in which pairwise constraints were treated as Bayesian priors. In [32], pairwise constraints were formed as an additional penalty term in the objective of spectral learning. Law *et al.* [36] proposed a graphical model that considers pairwise constraints as random variables; then an EM-based algorithm was applied to model the uncertainty of the constraints. Kulis *et al.* [35] further enhanced the performance of K-means clustering by learning a kernel matrix which incorporates the given pairwise constraints. In [61], an inductive matrix completion technique was employed to solve the constrained clustering problem, where data features were treated as side information.

The second group of semi-supervised clustering methods depend on distance metric learning (DML) techniques [60]. In a typical fashion, these methods first learn a distance metric with given pairwise constraints, such that the data pairs with must-link constraints have smaller distances than those pairs with cannot-link constraints. They then derive a linear transform from the learned distance metric and readily apply it to yield a new vector representation for the raw data points. The final data partition is simply obtained through running existing clustering algorithms over the transformed data representation. Various distance metric learning algorithms have been incorporated into semi-supervised clustering. Xing *et el.* [60] formulated distance metric learning as a PSD constrained convex programming problem. In [19], an information-theoretic method was exploited to learn a Mahalanobis distance function. Weinberger *et el.* [59] proposed a nearest-neighbor classifier to enforce that the data pairs from different classes are separated by a large margin. In [1], relevant component analysis was conducted to learn a distance metric by assigning larger weights to relevant dimensions and smaller weights to irrelevant dimensions, respectively. This work was further improved in [29] by simultaneously exploring both must-link and cannot-link constraints. In addition to the linear DML algorithms, a number of nonlinear DML algorithms have also been proposed to construct nonlinear mappings from the original data space [55, 49, 28, 12].

In recent years, an increasing amount of literature has begun to study the problem of evolving clustering. For instance, clustering results can be updated according to user's feedback [15, 7]. In particular, Cohn *et el.* [15] considered a scenario that users can iteratively provide different types of feedbacks regarding the clustering quality, and then developed an EM-like algorithm to update the distance metric towards achieving a higher-quality clustering. The algorithm proposed in [7] is a variant of the complete-link hierarchical clustering algorithm, which combined the feedback stemming from pairwise constraints to learn a new distance metric. Although the aforementioned algorithms can improve the clustering performance by leveraging user's feedback, they all suffer from a scalability issue since learning

[1]A probability simplex vector is a vector whose elements are non-negative and sum up to 1.

distance metrics is computationally expensive, especially in high dimensions. The main focus of evolutionary clustering [11, 13] is to learn a clustering that fits the current data well and does not shift dramatically from the historical data. Chakrabarti *et el.* [11] developed the evolutionary versions of both the K-means and agglomerative hierarchical clustering algorithms. Chi *et el.* [13] then extended this idea to develop the evolutionary spectral clustering algorithm. Besides, dynamic network clustering [11, 13, 52, 39, 33] was suggested to solve the community evolution problem when a network to be clustered changes continuously. In [52], a parameter-free algorithm called GraphScope was proposed to mine time-evolving graphs obeying the principle of Minimum Description Length (MDL). FacetNet [39] employed probabilistic community membership models to identify dynamic communities within a time-evolving graph. Kim and Han [33] further allowed a varying number of communities and presented a particle-and-density based algorithm to discover new communities or dissolve existing communities. Albeit looking similar, dynamic network clustering is different from the focus of this paper due to the following reasons: (i) dynamic network clustering algorithms only use links to guide clustering but ignore the important feature information; (ii) they rely on a large amount of link information to conduct clustering, while our studied dynamic clustering only requires a small number of pairwise constraints. Due to the flexibility of handling both data features and dynamic relationships, our proposed semi-supervised clustering approach better fits conventional clustering applications.

3. SEMI-SUPERVISED CLUSTERING WITH SEQUENTIAL CONSTRAINTS

In this section, we first present a general framework for semi-supervised clustering, followed by the proposed efficient algorithm for dynamic semi-supervised clustering.

3.1 Semi-Supervised Clustering

Let $\mathcal{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ be a set of n data points to be clustered, where each data point $\mathbf{x}_i \in \mathbb{R}^d,\ i \in [n]$ is a vector of d dimensions. Let $\mathcal{M}_t$ be the set of must-link constraints generated until time t, where each must-link pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t$ implies that $\mathbf{x}_i$ and $\mathbf{x}_j$ should be in the same cluster. Similarly, let $\mathcal{C}_t$ be the set of cannot-link constraints generated until time t, where each cannot-link pair $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t$ implies that $\mathbf{x}_i$ and $\mathbf{x}_j$ should belong to different clusters. For ease of presentation, we also define $\Omega_t = \mathcal{M}_t \cup \mathcal{C}_t$ to include all pairwise constraints generated until time t. Similar to most studies on data clustering, we assume that the number of clusters r is given a priori. Throughout this paper, we use a binary matrix $F \in \{0,1\}^{n\times r}$ to represent the result of partitioning n data points into r clusters, where $F_{ij} = 1$ indicates that $\mathbf{x}_i$ is associated with the j-th cluster. We further denote $\mathcal{F}$ as the set of all possible clustering results

$$\mathcal{F} = \{F \in \{0,1\}^{n\times r} \colon F_{*,i}^\top F_{*,j} = 0 \ \ \forall i \neq j,\ \sum F_{k,*} = 1\ \forall k\},$$

where $F_{k,*}$ and $F_{*,i}$ indicate the k-th row vector and i-th column vector of matrix F, respectively. Let $\kappa(\mathbf{x}, \mathbf{x}')$ be a kernel function used to measure the similarity between two data points $\mathbf{x}$ and $\mathbf{x}'$, and let $K = [\kappa(\mathbf{x}_i, \mathbf{x}_j)] \in \mathbb{R}_+^{n\times n}$ be the kernel matrix. The goal of semi-supervised learning is to find the clustering result that is consistent with both the kernel similarities in K and pairwise constraints in Ω_t. To measure the discrepancy between the kernel similarity K and a clustering result F, we define the distance between K and F as

$$d(K, F) = \sum_{i=1}^{n} F_{i,*}^\top K^{-1} F_{i,*} = \mathrm{tr}\left(F^\top K^{-1} F\right). \tag{1}$$

As indicated by the above measure, the smaller the distance $d(K, F)$, the better the consistency between the clustering result F and the similarity matrix K. We note that an alternative approach is to measure the distance by $\mathrm{tr}(F^\top L F)$, where $L = \mathrm{diag}(K\mathbf{1}) - K$ is the graph Laplacian [56].

To measure the inconsistency between the clustering result F and pairwise constraints, we introduce two loss functions, one for must-links and the other for cannot-links. More specifically, given a must-link $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t$, we define the loss function $\ell_-(F_{i,*}, F_{j,*})$ as

$$\ell_-(F_{i,*}, F_{j,*}) = \|F_{i,*} - F_{j,*}\|_2^2. \tag{2}$$

Likewise, given a cannot-link $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t$, we define the loss function $\ell_+(F_{i,*}, F_{j,*})$ as

$$\ell_+(F_{i,*}, F_{j,*}) = \|F_{i,*} + F_{j,*}\|_2^2. \tag{3}$$

We note that a loss function similar to (3) has been used in label propagation with cannot-links [44]. The justification of using loss function $\ell_+(\cdot,\cdot)$ for cannot-links is provided in the following proposition.

PROPOSITION 1. *Let $\mathbf{a} \in \mathbb{R}_+^d$ be a fixed vector. Let $\mathbf{b}^*$ be the minimizer to the following optimization problem*

$$\min_{\mathbf{b}\in\mathbb{R}_+^d, \|\mathbf{b}\|_q \le R} \ell_+(\mathbf{a}, \mathbf{b})$$

where $q \geq 1$. Then we have $\mathbf{b}^* \perp \mathbf{a}$.

As indicated by Proposition 1, by minimizing the loss function $\ell_+(F_{i,*}, F_{j,*})$, the resulting solution under no other constraints will satisfy $F_{j,*} \perp F_{i,*}$, implying that $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to different clusters.

Using the distance measure $d(K, F)$ and the loss functions ℓ_+ and ℓ_-, we can cast semi-supervised clustering into the following optimization problem

$$\begin{aligned} \min_{F\in\mathcal{F}} \quad & \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}_t} \ell_+(F_{i,*}, F_{j,*}) + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}_t} \ell_-(F_{i,*}, F_{j,*}) \\ \text{s.t.} \quad & d(F, K) \le \varepsilon, \end{aligned} \tag{4}$$

where the threshold ε decides the level of consistency between the clustering results and the kernel similarity.

The main challenge of dynamic semi-supervised clustering arises from the fact that pairwise constraints are sequentially updated over time. A naive approach is to solve the optimization problem in (4) from scratch whenever pairwise constraints are updated. A more efficient approach is to exploit the fact that only a small portion of pairwise constraints are updated at each time, leading to a small change in the clustering result. Based on this intuition, we can use the existing solution as an initial solution to the optimization problem with updated constraints. Despite the simplicity, this approach can significantly reduce the running time, and has been widely used in clustering social networks with dynamic updates [11, 13]. Although this simple approach reduces the number of iterations due to the appropriate initialization, it still needs to solve the optimization problem in (4),

which is computationally expensive when the number of data points n is very large. To address this challenging issue, we propose an efficient dynamic semi-supervised clustering algorithm that is highly efficient for clustering large-scale data sets.

3.2 Dynamic Semi-Supervised Clustering

The proposed algorithm is based on a key observation that the number of different clustering results F in the set $\Delta = \{F \in \mathcal{F} : d(K, F) \leq \varepsilon\}$ is not very large when ε is relatively small and the eigenvalues of K follow a skewed distribution.

To see this, we denote by $\lambda_1, \ldots, \lambda_n$ the eigenvalues of K ranked in descending orders, and $\mathbf{v}_1, \ldots, \mathbf{v}_n$ the corresponding eigenvectors. $\{\lambda_k\}$ follows a q-power law if there exists a constant c such that $\lambda_k \leq ck^{-q}$, where $q > 2$. The following lemma summarizes an important property of K when its eigenvalues follow a q-power law.

LEMMA 1. *Define $V = (\mathbf{v}_1, \ldots, \mathbf{v}_n)$ and $\xi = V^\top \mathbf{x}$ for a unit vector $\|\mathbf{x}\|_2 = 1$. If $\mathbf{x}^\top K^{-1}\mathbf{x} \leq \varepsilon$, we have*

$$\frac{\|\xi\|_1}{\|\xi\|_2} \leq \sqrt{\varepsilon n}\left(1 + \frac{2}{q-2}\right),$$

given that the eigenvalues of K follow a q-power law.

The above lemma shows that when the eigenvalues of K follow a power law, $V^\top \mathbf{x}$ is an ℓ_1 sparse vector if $\mathbf{x}^\top K^{-1}\mathbf{x} \leq \varepsilon$. This observation provides a key foundation for our analysis.

Define $\theta_n(\rho, r)$ to be the maximum number of partitions in Δ for r-way clustering such that the difference between any two partitions is at least ρ. The theorem below bounds $\theta_n(\rho, r)$.

THEOREM 1.

$$\theta_n(\rho, r) \leq \left[\frac{2n}{(r-1)s}\right]^{Cs(r-1)/(2\rho)}$$

where C is an universal constant and

$$s = \sqrt{\varepsilon n}\left(1 + \frac{2}{q-2}\right)$$

The proof of Lemma 1 and Theorem 1 are deferred to the Appendix.

Remark: Theorem 2 implies that when ρ is large enough, $\theta_n(\rho, r)$ is upper bounded by $O(n^{O(r\sqrt{\varepsilon})})$, which could be significantly less than the number of possible r-way clustering partitions r^n.

Based on the above results, there is a relatively small number of significantly different clustering results in the subspace Δ. Hence, to improve the computational efficiency of dynamic semi-supervised clustering, a natural thought is to pre-compute all the possible clustering results in Δ, and find the best clustering result in Δ that is consistent with most of the dynamically updated pairwise constraints. However, since the number of different clustering results in Δ is still large, it is computationally intractable to identify all of them.

To address this problem, we propose to construct a convex hull $\widetilde{\Delta} \subset \mathcal{F}$ to approximate the set Δ. The key advantage of using a convex hull approximation is that all the solutions in $\widetilde{\Delta}$ can be represented by convex combinations of $\widetilde{\Delta}$'s extreme points[2]. Thus, in order to find the best clustering result, we only need to compute the best combination weights of $\widetilde{\Delta}$'s extreme points. Since the number of combination weights to be determined is much smaller than the number of data points to be clustered, this enables us to compute combination weights in an extremely efficient way.

Specifically, the proposed clustering process is composed of two steps: an offline and an online step. In the offline step, we generate multiple partitions of the same dataset $\mathcal{X}$ and use such partitions to construct a convex hull $\widetilde{\Delta}$. In the online step, an efficient learning algorithm is developed to update the combination weights based on the newly received pairwise constraints. Below, we describe the two steps in detail.

3.2.1 Offline Step

In this step, we generate the convex hull $\widetilde{\Delta}$ using the technique of ensemble clustering [51]. The main idea behind ensemble clustering is to combine multiple partitions of a same dataset into a single data partition, hoping to exploit the strength of different clustering results, and compensate for their limitations.

According to [26], multiple partitions of a same dataset can be generated by (i) applying different clustering algorithms [24], (ii) using the same algorithm with different initializations and parameters [53, 23], (iii) employing different pre-processing and feature extraction mechanisms [9], and (iv) exploring different subsets of features [24, 21]. In order to efficiently generate ensemble partitions for large-scale data sets, we employee the last approach by first randomly selecting m ($m \ll n$) different subsets of features in $\mathcal{X}$, followed by applying the approximate kernel K-means algorithm [14] to each feature subset. Since the cluster labels of such ensemble partitions can be arbitrary, we then apply the Hungarian algorithm [34] to realign their labels. We denote by $\mathcal{P} = \{P_1, P_2, \ldots, P_m\}$ the m realigned ensemble partitions, where each partition $P_i \in \mathcal{F},\ i \in [m]$ divides $\mathcal{X}$ into r disjoint subsets.

The following proposition shows that the ensemble partitions $P_1, \ldots, P_m$ are the m extreme points of the convex hull $\text{conv}\{P_1, \ldots, P_m\}$.

PROPOSITION 2. *The convex hull*

$$\begin{aligned}\widetilde{\Delta} &= \text{conv}\{P_1, \ldots, P_m\} \\ &= \{\gamma_1 P_1 + \cdots + \gamma_m P_m : \sum_{i=1}^{m} \gamma_i = 1 \text{ and } \gamma_i \geq 0,\ i \in [m]\}\end{aligned}$$

has m extreme points, with each of them being equal to $P_i,\ i \in [m]$.

Proposition 2 shows that the data partitions $P_i,\ i \in [m]$, are not interior points of any line segments lying entirely in the convex hull $\widetilde{\Delta}$. Hence, all the solutions in $\widetilde{\Delta}$ can be represented by convex combinations of the m ensemble partitions in the set $\mathcal{P}$.

We claim that the convex hull $\widetilde{\Delta}$ is a good approximation of the set $\Delta = \{F \in \mathcal{F} : d(K, F) \leq \varepsilon\}$, due to the following two reasons:

[2]An extreme point of a convex set $\mathcal{S}$, is a point $x \in \mathcal{S}$, with the property that if $x = \theta y + (1-\theta)z$ with $y, z \in \mathcal{S}$ and $\theta \in [0, 1]$, then $y = x$ or $z = x$.

1. The ensemble partitions are generated by exploring a kernel k-means algorithm, which is known to be closely related to spectral clustering [20]. Note that the clustering results in Δ are agreeable to all the d features of data points, they should also agree with any subsets of data features. Therefore, all the ensemble partitions should satisfy the condition

$$d(K, P_i) \leq \phi, \quad \forall i \in [m],$$

where ϕ is a constant that is larger than ε. This implies that Δ and $\widetilde{\Delta}$ should have a large overlap.

2. According to the widely applied median partition based ensemble clustering [51, 54, 25], the optimal data partition P^* should have low divergence from all the partitions in the clustering ensemble. Since all the ensemble partitions are the extreme points of the convex hull $\widetilde{\Delta}$, P^* should lie in the inner space spanned by $\widetilde{\Delta}$ and it can be uniquely represented by a convex combination of ensemble partitions.

By exploiting the convex hull as a search space, we are able to cast the problem of clustering n data points into the problem of learning a m-dimensional probability simplex vector $\boldsymbol{\gamma}$. In more detail, other than directly computing the best F, we solve the following convex optimization problem to find the optimal $\boldsymbol{\gamma}$

$$\begin{aligned} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^m} \quad & \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t} \ell_+(F_{i,*}, F_{j,*}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t} \ell_-(F_{i,*}, F_{j,*}) \\ \text{s.t.} \quad & \boldsymbol{\gamma}^\top \mathbf{1} = 1, \quad F = \sum_{i=1}^m \gamma_i P_i \end{aligned} \tag{5}$$

Since m is a small number that is much less than the data size n, the optimization problem in (5) can be solved in an extremely efficient way[3]. When no pairwise constraints is involved, the problem of semi-supervised clustering reduces to a standard ensemble clustering problem. The clustering result in terms of the probability simplex vector, denoted by $\boldsymbol{\gamma}^0$, can be simply set to $(\frac{1}{m}, \ldots, \frac{1}{m})$. In the following, we introduce the online step for efficiently updating data partitions when pairwise constraints are generated sequentially.

3.2.2 Online Step

In this subsection, we develop an *online* step to efficiently update the probability simplex vector given new pairwise constraints. To simplify the presentation, we divide $\mathcal{M}_t$, the set of must-link constraints received until time t, into two subsets: $\mathcal{M}_t^a$ that includes all the must-link constraints received before time t and $\mathcal{M}_t^b$ that includes the new must-link constraints added at time t. Similarly, we divide the cannot-link set $\mathcal{C}_t$ into $\mathcal{C}_t^a$ and $\mathcal{C}_t^b$. We also denote by $\boldsymbol{\gamma}^1, \ldots, \boldsymbol{\gamma}^t$ the sequence of probability simplex vectors computed based on the updated constraints.

Using the above notations, we rewrite the optimization problem in (5) as

$$\begin{aligned} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^m} \quad & \mathcal{L}_t^a(F(\boldsymbol{\gamma})) + \mathcal{L}_t^b(F(\boldsymbol{\gamma})) \\ \text{s.t.} \quad & \boldsymbol{\gamma}^\top \mathbf{1} = 1, \quad F = \sum_{i=1}^m \gamma_i P_i, \end{aligned} \tag{6}$$

[3]We note that the ensemble partitions are computed offline and therefore do not affect the efficiency of online computation, which is the main concern of this work.

Algorithm 1 The projection of a vector onto the probability simplex [58]

Input: a vector $\mathbf{v} \in \mathbb{R}^m$ to be projected

1: Sort $\mathbf{v}$ into $\mathbf{w}$: $w_1 \geq w_2 \geq \cdots \geq w_m$
2: Find $k = \max\{j \in [m] : w_j - \frac{1}{j}(\sum_{i=1}^k w_i - 1) > 0\}$
3: Compute $\theta = \frac{1}{k}(\sum_{i=1}^k w_i - 1)$

Return: $\mathbf{u} \in \mathbb{R}^m$ s.t. $\mathbf{u}_i = \max(\mathbf{w}_i - \theta,\ 0),\ i \in [m]$

where

$$\begin{aligned} \mathcal{L}_t^a(F) &= \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t^a} \ell_+(F_{i,*}, F_{j,*}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t^a} \ell_-(F_{i,*}, F_{j,*}) \\ \mathcal{L}_t^b(F) &= \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t^b} \ell_+(F_{i,*}, F_{j,*}) + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t^b} \ell_-(F_{i,*}, F_{j,*}). \end{aligned}$$

Since $\boldsymbol{\gamma}^{t-1}$ is the minimizer of the objective $\mathcal{L}_t^a(F(\boldsymbol{\gamma}))$, we can approximate $\mathcal{L}_t^a(F(\boldsymbol{\gamma}))$ as

$$\mathcal{L}_t^a(F(\boldsymbol{\gamma})) \approx \mathcal{L}_t^a(F(\boldsymbol{\gamma}^{t-1})) + \lambda \|\boldsymbol{\gamma} - \boldsymbol{\gamma}^{t-1}\|_2^2.$$

As a result, the optimization problem in (6) is further simplified as

$$\begin{aligned} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^m} \quad & \mathcal{L}_t^b(F(\boldsymbol{\gamma})) + \lambda \|\boldsymbol{\gamma} - \boldsymbol{\gamma}^{t-1}\|_2^2 \\ \text{s. t.} \quad & \boldsymbol{\gamma}^\top \mathbf{1} = 1, \quad F = \sum_{i=1}^m \gamma_i P_i, \end{aligned} \tag{7}$$

where parameter λ is introduced to balance between two objectives, *i.e.*, ensuring that the learned $\boldsymbol{\gamma}$ is not far away from $\boldsymbol{\gamma}^{t-1}$, and also consistent with the new pairwise constraints. Compared to (6), the main advantage of problem (7) is that it only involves the new constraints that are added to the system at time t and does not need to store and work with the constraints received before time t.

In the following, we discuss how to efficiently update the probability simplex vector given the new pairwise constraints. By incorporating F into the optimization problem, we rewrite problem (7) as

$$\begin{aligned} \min_{\boldsymbol{\gamma} \in \mathbb{R}_+^m} \quad & f(\boldsymbol{\gamma}) = \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{M}_t^b} || \sum_{k=1}^m \boldsymbol{\gamma}_k [P_k(i,:) - P_k(j,:)] ||_2^2 \\ & + \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{C}_t^b} || \sum_{k=1}^m \boldsymbol{\gamma}_k [P_k(i,:) + P_k(j,:)] ||_2^2 \\ & + \lambda ||\boldsymbol{\gamma} - \boldsymbol{\gamma}^{t-1}||_2^2 \\ \text{s.t.} \quad & 1^\top \boldsymbol{\gamma} = 1, \end{aligned} \tag{8}$$

where $P_k(i,:)$ and $P_k(j,:)$ represent the i-th and the j-th row vector of the matrix P_k, respectively. The optimization problem (8) can be efficiently solved by a gradient descend method. Specifically, in the q-th iteration, we update the probability simplex $\boldsymbol{\gamma}$ by

$$\boldsymbol{\gamma}^{q+1} = P_{\widetilde{\Delta}}(\boldsymbol{\gamma}^q - \eta \ \nabla f(\boldsymbol{\gamma}^q)),$$

where η is a step size and $P_{\widetilde{\Delta}}$ is an efficient algorithm that projects a m-dimensional vector onto the probability simplex in $O(m \log m)$ time, as described in Algorithm 1.

Note that $\nabla f(\boldsymbol{\gamma}^q)$ has a closed-form solution as

$$\nabla f(\boldsymbol{\gamma}^q) = 2 [\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}_t^b} \boldsymbol{\gamma}^q U^{(ij)}U^{(ij)\top} + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}_t^b} \boldsymbol{\gamma}^q V^{(ij)}V^{(ij)\top} + \lambda (\boldsymbol{\gamma}^q - \boldsymbol{\gamma}^{t-1})], \quad (9)$$

where $U^{(ij)}$ and $V^{(ij)}$ are two $m \times r$ matrices, satisfying

$$U^{(ij)}(k,:) = P_k(i,:) - P_k(j,:)$$

and

$$V^{(ij)}(k,:) = P_k(i,:) + P_k(j,:).$$

Then the probability simplex can be updated via

$$P_{\tilde{\Delta}} \{ \boldsymbol{\gamma}^q [I_m - 2\eta (\sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}_t^b} U^{(ij)}U^{(ij)\top} + \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}_t^b} V^{(ij)}V^{(ij)\top})] - 2\lambda\eta (\boldsymbol{\gamma}^q - \boldsymbol{\gamma}^{t-1})\}, \quad (10)$$

where I_m is the $m \times m$ identity matrix. Since the matrices $U^{(ij)}U^{(ij)\top}$ and $V^{(ij)}V^{(ij)\top}$ can be precomputed offline, we can efficiently update $\boldsymbol{\gamma}$ using equation (10).

Space-Efficient Relaxation: Despite low time complexity, the updating scheme (10) suffers from a large space complexity to store all the matrices $U^{(ij)}U^{(ij)\top}$ and $V^{(ij)}V^{(ij)\top}$. We now discuss how to reduce the storage cost by relaxing the optimization procedure.

Note that the k-th row of the matrix $U^{(ij)}$ should be either of these two cases: (i) containing all zero entries if the ensemble partition P_k put object i and object j in the same cluster, or (ii) containing one positive entry ($= 1$) , and one negative entry ($= -1$) if the ensemble partition P_k put object i and object j in different clusters. Then the diagonal elements of the matrix $U^{(ij)}U^{(ij)\top}$ either equal 0 or equal a positive value ($= 2$). Thus the matrix

$$I_m - 2\eta \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{M}_t^b} U^{(ij)}U^{(ij)\top}$$

essentially assigns less weight to the ensemble partitions that mistakenly put the object i and object j in different clusters when they share a must-link connection. Likewise, the matrix

$$I_m - 2\eta \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in\mathcal{C}_t^b} V^{(ij)}V^{(ij)\top}$$

essentially assigns less weight to the ensemble partitions that mistakenly put the object i and object j in the same cluster when they share a cannot-link constraint. After updating $\boldsymbol{\gamma}^{q+1}$ from $\boldsymbol{\gamma}^q$, the ensemble partitions that are consistent with the new pairwise constraints are assigned larger weights, while the ensemble partitions that are not consistent with the new pairwise constraints are assigned smaller weights. This leads to a relaxed updating procedure[4]

$$P_{\tilde{\Delta}} \{(1 - 2\lambda\eta) \boldsymbol{\gamma}^q + 2\lambda\eta \boldsymbol{\gamma}^{t-1} - C\eta \sum_{(\mathbf{x}_i,\mathbf{x}_j)\in \{\mathcal{M}_t^b\cup\mathcal{C}_t^b\}} \mathbf{e}^{(ij)}\}, \quad (11)$$

where $\mathbf{e}^{(ij)}$ is an m-dimensional vector with the k-th element equaling 0 if the ensemble partition P_k is consistent with the pairwise constraints $(\mathbf{x}_i, \mathbf{x}_j)$, and 1 otherwise. The parameter $C > 0$ is introduced to ensure that the third term in (11) is comparable with the first two terms.

Given the learned probability simplex $\boldsymbol{\gamma}$, we can generate a soft labeling matrix as a linear combination of the ensemble partitions

$$P = \boldsymbol{\gamma}_1 P_1 + \boldsymbol{\gamma}_2 P_2 + \ldots + \boldsymbol{\gamma}_m P_m.$$

Then the hard partition can be easily obtained by applying the efficient K-means clustering algorithm [8] to P or rounding, i.e., assigning the i-th data point to the k-th cluster if P_{ik} is the largest entry in the i-th row of P.

The time complexity to update the probability simplex vector is $O(pm^2)$, where p is the number of pairwise constraints added at time t. Given the updated $\boldsymbol{\gamma}^t$, the time complexity to generate a hard partition is $O(mnr)$. Since generally speaking, both m and r are much smaller than n, the total time complexity of the proposed algorithm is $O(n)$, which cannot be further improved since all n data points must be go through at least once for assignment.

4. EXPERIMENTS

In this section, we empirically demonstrate that our proposed semi-supervised clustering algorithm is both efficient and effective.

4.1 Datasets

Four real-world benchmark datasets with varied sizes are used in our experiments, which are:

- **COIL20** [46], a dataset containing 20 objects with $1,440$ images in total. Each image is represented by a 1024-dimensional vector.
- **USPS** [30], a widely used handwritten digits dataset including $9,298$ handwritten images. Each image is represented by a 256-dimensional vector that belongs to one of 10 classes.
- **Covtype**[5], a dataset used to predict forest cover types using cartographic variables. This dataset consists of $581,012$ records belonging to seven cover type classes, *i.e.*, spruce/fir, lodgepole pine, ponderosa pine, cottonwood/willow, aspen, douglas-fir, and krummholz.
- **MNIST8m** [43], a dataset artificially enlarged from the MNIST handwritten digits dataset[6]. It contains a total of $8,100,000$ samples that belong to 10 classes.

4.2 Parameter Selection

In order to generate m ensemble partitions, we need to randomly sample $\tilde{d}$ out of d features in each time. Two criteria are adopted in determining the value of $\tilde{d}$. First, $\tilde{d}$ should be small enough to make ensemble partitions diverse. Second, $\tilde{d}$ should be reasonably large to generate reliable ensemble partitions since the quality of the starting point $\boldsymbol{\gamma}^0$ depends on the quality of the ensemble partitions. In our experiments, we set $\tilde{d} = \lceil d/20 \rceil$.

In addition, as a key factor which affects the performance of the proposed clustering algorithm, the number of ensemble partitions m introduces a trade-off between the clus-

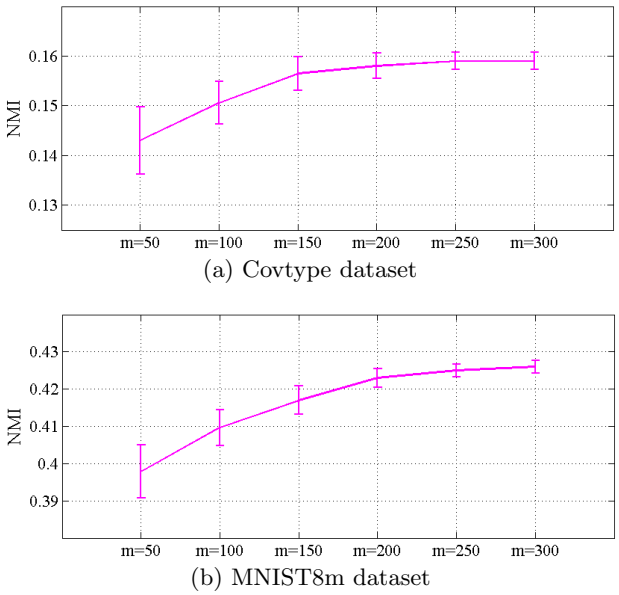
[4]We note that $U^{(ij)}U^{(ij)\top}$ and $V^{(ij)}V^{(ij)\top}$ may not be diagonal matrices. Thus the relaxation that only considers their diagonal elements can lead to information loss, a problem that will be investigated in our future work.

[5]`https://archive.ics.uci.edu/ml/datasets/Covertype`

[6]`http://yann.lecun.com/exdb/mnist/`

tering quality and efficiency. As m increases, the clustering quality tends to improve at the cost of increased computing time. In the following, we analyze how the clustering performance will be influenced by m. To this end, we conduct the experiments on the two largest datasets, Covtype and MNIST8m. On both of the two datasets, we begin with the unsupervised data partition $\boldsymbol{\gamma}^0$ generated from $m = \{50,\ 100,\ 150,\ 200,\ 250,\ 300\}$ different ensemble partitions. Then we randomly generate 500 pairwise constraints based on the ground truth categorizations. For each m, we apply the proposed algorithm to update the data partition and use normalized mutual information (NMI for brevity) [16] to measure the coherence between the updated clustering result and the ground truth categorizations. This experiment is repeated ten times, and the clustering performance NMI, computed as the average over the ten trials, is plotted in Figure 1.

As expected, the clustering performance NMI keeps increasing when m becomes larger. The performance gain is due to the fact that a larger m indicates not only a larger search space but also a larger overlap between Δ and the convex hull $\tilde{\Delta}$. In addition, a larger number of ensemble partitions usually provide a more diverse clustering result, leading to a higher chance of finding the data partitions that are consistent with most or even all of the pairwise constraints. We also notice that the clustering performance NMI of the proposed algorithm gradually stabilizes as m increases to 300. This is not surprising, since when the number of ensemble partitions is already large enough, adding more partitions cannot provide more information because the new partitions are likely to coincide with some existing ensemble partitions. Such observations hence offer the guidance to appropriately choose m: on one hand, m should be reasonably large to provide a diverse and sufficiently large search space; on the other hand, m should be relatively small to reduce the computational cost. We set $m = 300$ throughout all the remaining experiments.



(a) Covtype dataset

(b) MNIST8m dataset

Figure 1: The clustering performance normalized mutual information (NMI) vs. the number of ensemble partitions $m = \{50,\ 100,\ 150,\ 200,\ 250,\ 300\}$.

4.3 Experimental Results

To examine the effectiveness and efficiency of the proposed semi-supervised clustering algorithm, we compare it against the following distance metric learning and constrained clustering algorithms: (a) **PGDM**, the probabilistic global distance metric learning algorithm [60], (b) **LMNN**, the large margin nearest-neighbor classifier [59], (c) **ITML**, the information-theoretic metric learning algorithm [19], (d) **RCA**, the relevant component analysis based metric learning algorithm [1], (e) **DCA**, the discriminative component analysis based metric learning algorithm [29], (f) **CCSKL**, the constrained clustering algorithm via spectral kernel learning [38], and (g) **PMMC**, the pairwise constrained maximum margin clustering algorithm [62]. We refer to the proposed clustering algorithm as **S**emi-supervised **C**lustering with **S**equential **C**onstraints, or the abbreviation **SCSC**.

In our experiments, we begin with u randomly generated pairwise constraints, denoted by the tier t_1. In each of the following tiers, another set of u randomly sampled pairwise constraints are generated, and all the compared semi-supervised clustering algorithms are called to update their data partitions based on the newly generated pairwise constraints. Specifically, we rerun all the compared algorithms by adding the new pairwise constraints into the old ones. We repeat such steps from tier t_1 to tier t_5, finally resulting in a total of $5u$ randomly sampled pairwise constraints. Since the MNIST8m and Covtype datasets are much larger than the COIL20 and USPS datsets, we set $u = 500$ for the former two datasets, and $u = 100$ for the latter two datsets. All the experiments are performed on a Linux machine with Intel Xeon 2.4 GHz processor and 64 GB of main memory. Each experiment is repeated ten times, and the average clustering performance NMI and the average running time are reported. We mark the running time as N/A if an algorithm cannot converge to yield meaningful data partitions within 2 hours.

Figure 2 displays the curves of the clustering performance for all the referred semi-supervised clustering algorithms, where we exclude the four baseline algorithms (**PGDM**, **LMNN**, **CCSKL**, and **PMMC**) on the two large datasets, *i.e.*, MNIST8m and Covtype, since the data partitions cannot be updated by such algorithms within 2 hours. In comparison to the other competing algorithms, the proposed dynamic semi-supervised clustering algorithm **SCSC** accomplishes the best performance on all the four datasets. It is important that the proposed **SCSC** algorithm outperforms all the competing algorithms from the very beginning, *i.e.*, when only a small number of pairwise constraints are given. The reason is that by generating a convex hull from a set of ensemble partitions, we actually reduce the possible search space dramatically and all the inner points in that convex hull can map to reasonably good data partitions. Also, since the starting point $\boldsymbol{\gamma}^0$ is produced by incorporating the strengths of multiple ensemble partitions, $\boldsymbol{\gamma}^0$ should already be close to the optimal solution. Therefore, a simple local search should be good enough to recover the optimal partition. On the negative side, the variance of the proposed **SCSC** algorithm is relatively large, especially comparing with some distance metric learning algorithms such as **RCA**, **DCA** and **ITML**. We conjecture that the large

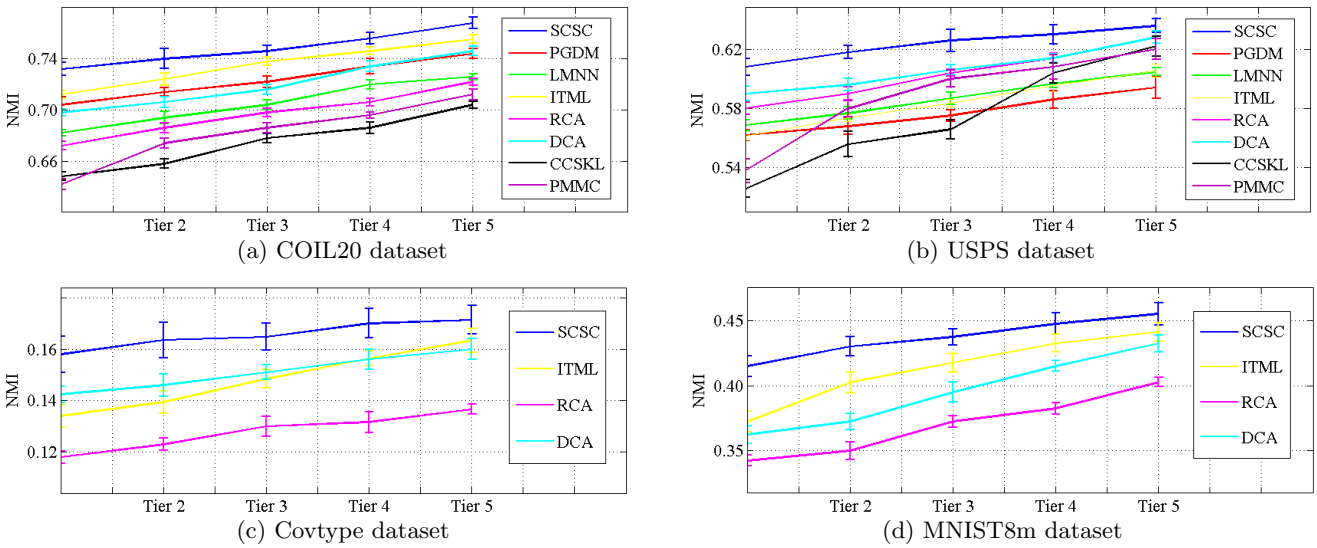


Figure 2: Average clustering performance NMI achieved by the proposed dynamic semi-supervised clustering algorithm SCSC and the competing algorithms including PGDM [60], LMNN [59], ITML [19], RCA [1], DCA [29], CCSKL [38], and PMMC [62] from tier t_1 to t_5 on four datasets.

Table 1: Average running time (in seconds) for updating the data partitions in each tier. N/A means that the clustering task cannot be completed by the corresponding algorithm within 2 hours.

Running Time (sec)	SCSC	PGDM	LMNN	ITML	RCA	DCA	CCSKL	PMMC
COIL20	**0.2**	74	131	15	0.8	0.4	56	264
USPS	**0.5**	304	514	20	1.6	0.9	373	791
Covtype	**2.3**	N/A	N/A	232	11	6.7	N/A	N/A
MNIST8m	**5.2**	N/A	N/A	4,727	28	16	N/A	N/A

variance may be caused by the randomness in generating ensemble partitions. In our future work, we will investigate and endeavor to address the problem of generating less uncertain ensemble partitions.

Finally, we evaluate the computational efficiency of the proposed **SCSC** algorithm. Table 1 shows that the updating procedure of **SCSC** is extremely efficient. In particular, **SCSC** is able to update the partitioning results of more than 8 million samples in about 5 seconds.

5. CONCLUSIONS

In this paper, we proposed a dynamic semi-supervised clustering algorithm which can efficiently update clustering results given newly received pairwise constraints. The key idea is to cast the dynamic clustering process into a search problem over a feasible clustering space that is defined as a convex hull generated by multiple ensemble partitions. Since any inner point of the convex hull can be uniquely represented by a probability simplex vector, the dynamic semi-supervised clustering problem can be reduced to the problem of learning a low-dimensional vector. Given a set of sequentially received pairwise constraints, we devised an updating scheme to update the data partition in an extremely efficient manner. Our empirical studies conducted on several real-world datasets confirmed both the effectiveness and efficiency of the proposed algorithm.

6. REFERENCES

[1] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *JMLR*, 6:937–965, 2005.

[2] S Basu, A Banerjee, and R J. Mooney. Semi-supervised clustering by seeding. In *ICML*, pages 27–34, 2002.

[3] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *KDD*, pages 59–68, 2004.

[4] R. Bekkerman and M. Sahami. Semi-supervised clustering using combinatorial MRFs. In *ICML Workshop on Learning in Structured Output Spaces*, 2006.

[5] S. Bhatia and J. Deogun. Conceptual clustering in information retrieval. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 28(3):427–436, 1998.

[6] M Bilenko, S Basu, and R J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, 2004.

[7] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, pages 39–48, 2003.

[8] D Cai. Litekmeans: the fastest matlab implementation of kmeans. *Software available at: http://www. zjucadcg. cn/dengcai/Data/Clustering. html*, 2011.

[9] Marine Campedel, Ivan Kyrgyzov, and Henri Maitre. Consensual clustering for unsupervised feature selection. application to spot5 satellite image indexing. In *JMLR: Workshop and Conference Proceedings*, number 4, pages 48–59, 2008.

[10] Claudio Carpineto and Giovanni Romano. A lattice conceptual clustering system and its application to browsing retrieval. volume 24, pages 95–122, 1996.

[11] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *KDD*, pages 554–560, 2006.

[12] Shiyu Chang, Charu C. Aggarwal, and Thomas S. Huang. Learning local semantic distances with limited supervision. In *ICDM*, pages 70–79, 2014.

[13] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *KDD*, pages 153–162, 2007.

[14] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain. Approximate kernel k-means: solution to large scale kernel clustering. In *KDD*, pages 895–903, 2011.

[15] D. Cohn, R. Caruana, and A. McCallum. Semi-supervised clustering with user feedback. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*, 4(1):17–32, 2003.

[16] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 2006.

[17] Ian Davidson and S. S. Ravi. Agglomerative hierarchical clustering with constraints: Theoretical and empirical results. In *PKDD*, pages 59–70, 2005.

[18] Ian Davidson and S. S. Ravi. Clustering with constraints: Feasibility issues and the k-means algorithm. In *SDM*, 2005.

[19] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. In *ICML*, pages 209–216, 2007.

[20] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *KDD*, pages 551–556, 2004.

[21] C. Domeniconi and M. Al-Razgan. Weighted cluster ensembles: Methods and analysis. *TKDD*, 2(4), 2009.

[22] W. Fan and A. Bifet. Mining big data: current status, and forecast to the future. *ACM SIGKDD Explorations Newsletter*, 14(2):1–5, 2013.

[23] X. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *ICML*, pages 186–193, 2003.

[24] X. Fern and C. E. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *ICML*, 2004.

[25] Lucas Franek and Xiaoyi Jiang. Ensemble clustering by means of clustering embedding in vector spaces. *Pattern Recognition*, 47(2):833–842, 2014.

[26] Ana L. N. Fred and Anil K. Jain. Combining multiple clusterings using evidence accumulation. *PAMI*, 27(6):835–850, 2005.

[27] Hichem Frigui and Raghu Krishnapuram. A robust competitive clustering algorithm with applications in computer vision. *PAMI*, 21(5):450–465, 1999.

[28] T. Hertz, A. Hillel, and D. Weinshall. Learning a kernel function for classification with small training samples. In *ICML*, pages 401–408. ACM, 2006.

[29] S.C.H. Hoi, W. Liu, M.R. Lyu, and W.Y. Ma. Learning distance metrics with contextual constraints for image retrieval. In *CVPR*, pages 2072–2078, 2006.

[30] J.J. Hull. A database for handwritten text recognition research. *PAMI*, 16(5):550–554, 1994.

[31] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[32] S D. Kamvar, D Klein, and C D. Manning. Spectral learning. In *IJCAI*, pages 561–566, 2003.

[33] Min-Soo Kim and Jiawei Han. A particle-and-density based evolutionary clustering method for dynamic networks. *PVLDB*, 2(1):622–633, 2009.

[34] H. W. Kuhn and Bryn Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

[35] B. Kulis, S. Basu, I. S. Dhillon, and R. J. Mooney. Semi-supervised graph clustering: a kernel approach. *Machine Learning*, 74(1):1–22, 2009.

[36] M Law, A P. Topchy, and A K. Jain. Model-based clustering with probabilistic constraints. In *SDM*, 2005.

[37] Q Li and B Kim. Clustering approach for hybrid recommender system. In *Web Intelligence*, pages 33–38, 2003.

[38] Z. Li and J. Liu. Constrained clustering by spectral kernel learning. In *ICCV*, pages 421–427, 2009.

[39] Yu-Ru Lin, Y. Chi, S. Zhu, H. Sundaram, and B. L. Tseng. Facetnet: a framework for analyzing communities and their evolutions in dynamic networks. In *WWW*, pages 685–694, 2008.

[40] W. Liu, J. He, and S. Chang. Large graph construction for scalable semi-supervised learning. In *ICML*, pages 679–686, 2010.

[41] W. Liu, J. Wang, and S. Chang. Robust and scalable graph-based semisupervised learning. *Proceedings of the IEEE*, 100(9):2624–2638, 2012.

[42] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *SIGIR*, pages 186–193, 2004.

[43] G. Loosli, S. Canu, and L. Bottou. Training invariant support vector machines using selective sampling. *Large Scale Kernel Machines*, pages 301–320, 2007.

[44] Z. Lu and M. Á. Carreira-Perpiñán. Constrained spectral clustering through affinity propagation. In *CVPR*, 2008.

[45] Z. Lu and T. K. Leen. Semi-supervised learning with penalized probabilistic clustering. In *NIPS*, 2004.

[46] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). Technical report, Technical Report CUCS-005-96, 1996.

[47] Y. Plan and R. Vershynin. Robust 1-bit compressed sensing and sparse logistic regression: A convex programming approach. *CoRR*, abs/1202.1212, 2012.

[48] A. Rajaraman and J. D. Ullman. *Mining of massive datasets*, volume 77. Cambridge University Press Cambridge, 2012.

[49] C. Shen, J. Kim, L. Wang, and A. Hengel. Positive semidefinite metric learning with boosting. In *NIPS*, pages 1651–1659, 2009.

[50] N. Shental, A. Bar-Hillel, T. Hertz, and D. Weinshall. Computing gaussian mixture models with em using equivalence constraints. In *NIPS*, 2003.

[51] Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *JMLR*, 3:583–617, 2002.
[52] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *KDD*, pages 687–696, 2007.
[53] Alexander P. Topchy, Anil K. Jain, and William F. Punch. A mixture model for clustering ensembles. In *SDM*, 2004.
[54] Alexander P. Topchy, Anil K. Jain, and William F. Punch. Clustering ensembles: Models of consensus and weak partitions. *PAMI*, 27(12):1866–1881, 2005.
[55] L. Torresani and K. Lee. Large margin component analysis. In *NIPS*, pages 1385–1392, 2006.
[56] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.
[57] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *ICML*, pages 577–584, 2001.
[58] Weiran Wang and Miguel Á. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *CoRR*, abs/1309.1541, 2013.
[59] K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006.
[60] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *NIPS*, 15:505–512, 2002.
[61] J. Yi, L. Zhang, R. Jin, Q. Qian, and A. K. Jain. Semi-supervised clustering by input pattern assisted pairwise similarity matrix completion. In *ICML*, pages 1400–1408, 2013.
[62] H. Zeng and Y. Cheung. Semi-supervised maximum margin clustering with pairwise constraints. *IEEE Trans. Knowl. Data Eng.*, 24(5):926–939, 2012.

APPENDIX

A. PROOF OF LEMMA 1

Note that λ_i and $\mathbf{v}_i$, $i \in [n]$ are the eigenvalues and eigenvectors of the kernel matrix K. Since $\xi = V^\top \mathbf{x}$, we have

$$\mathbf{x}^\top K^{-1}\mathbf{x} = \sum_{i=1}^{N} \lambda_i^{-1}\xi_i^2 \le \varepsilon,$$

which implies

$$\xi_k \le \sqrt{c\varepsilon}k^{-q/2},\ k = 1, \ldots, N.$$

Thus, we have

$$\|\xi\|_1 \le \sqrt{c\varepsilon}\sum_{k=1}^{N} k^{-q/2} \le \sqrt{c\varepsilon}\left(1 + \frac{2}{q-2}\right).$$

We complete the proof by using the facts $\|\mathbf{x}\|_2 = 1$ and $c \le n$.

B. PROOF OF THEOREM 1

In order to show that the number of significantly different partitions in Δ is small, we first consider a simple case where the number of classes is 2. In this case, we can simplify the domain Δ as

$$\Delta_2 = \left\{\mathbf{v} \in \{-1, +1\}^n : \mathbf{v}^\top K^{-1}\mathbf{v} \le n\varepsilon\right\}.$$

We define $\theta_n(\rho)$ the maximum number of partitions in Δ_2 such that the difference between any two partition vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is at least ρ. The theorem below bounds $\theta_n(\rho)$.

THEOREM 2.

$$\theta_n(\rho) \le \left(\frac{2n}{s}\right)^{Cs/[2\rho]}$$

where C is an universal constant and

$$s = \sqrt{\varepsilon n}\left(1 + \frac{2}{q-2}\right) \tag{12}$$

PROOF. We first notice that $\|\mathbf{v}\|_2 = \sqrt{n}$. As the first step, we relax Δ_2 into Δ_2' as follows

$$\Delta_2' = \left\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \le \sqrt{n},\ \mathbf{x}^\top K^{-1}\mathbf{x} \le n\varepsilon\right\}$$

Define $\gamma_n(\Delta_2', \delta)$ the maximum number of vectors such that the distance between any two vectors is at least δ. Since the distance between any two partitions that differs by at least ρ entries is at least $2\sqrt{\rho}$, we have

$$\theta_n(\rho) \le \gamma_n\left(\Delta_2', 2\sqrt{\rho}\right).$$

Using the result of Lemma 1 and the covering number theorem [47] to bound, we have

$$\gamma_n\left(\Delta_2', 2\sqrt{\rho}\right) \le \exp\left(\frac{Cs}{2\rho}\log\frac{2n}{s}\right) = \left(\frac{2n}{s}\right)^{Cs/[2\rho]},$$

where s is defined in (12). □

The above result for two-way clustering can be easily extended into multiple-way clustering.

C. PROOF OF PROPOSITION 2

We prove Proposition 2 by contradiction. Suppose at least one ensemble partition P_i is not the extreme point of the convex hull $\tilde{\Delta} = \text{conv}\{P_1, \ldots, P_m\}$, then P_i should be an inner point of $\tilde{\Delta}$. According to the definition of extreme points, it is evident that all the extreme points of $\tilde{\Delta}$ should be feasible partitions in the set $\mathcal{F}$. We denote them as $F_1, \ldots, F_l \in \mathcal{F}$. Therefore, P_i can be represented by a convex combination of $F_1, \ldots, F_l$:

$$P_i = \sum_{j=1}^{l} \lambda_j F_j,$$

where $\boldsymbol{\lambda}$ is a l-dimensional probability simplex vector. Since each row of P_i contains only one 1 and all the other elements are 0, λ_j should be equal to 0 for any $F_j \ne P_i$. Otherwise P_i cannot be equal to $A = \sum_{j=1}^{l} \lambda_j F_j$ since there must exist at least one index (a, b) such that $P_i(a, b) = 0$ while $A(a, b) > 0$. Since $P_i \ne F_j, \forall j = 1, \ldots, l$, we have $\boldsymbol{\lambda} = 0^l$, which contradicts with the constraint $\mathbf{1}^\top\boldsymbol{\lambda} = 1$. This verifies that P_i is not a convex combination of other extreme points.

Since the number of the extreme points in $\tilde{\Delta}$ is at most m and each partition $P_i,\ i \in [m]$, is not a convex combination of the remaining $m-1$ ensemble partitions, the convex hull $\tilde{\Delta}$ has exactly m extreme points with each of them equaling to $P_i,\ i \in [m]$.