

# Fast indefinite multi-point (IMP) clustering

S. Chandrasekaran<sup>1</sup> · A. Rajagopal<sup>1</sup>

Received: 24 March 2015 / Accepted: 16 April 2016 / Published online: 29 April 2016  
© Springer-Verlag Italia 2016

**Abstract** A new class of objective functions and an associated fast descent algorithm that generalizes the  $K$ -means algorithm is presented. The algorithm represents clusters as unions of Voronoi cells and an explicit, but efficient, combinatorial search phase enables the algorithm to escape many local minima with guaranteed descent. The objective function has explicit penalties for gaps between clusters. Numerical experiments are provided.

**Keywords** Clustering · Monotonic descent · Combinatorial search · Voronoi cells

**Mathematics Subject Classification** Primary: 62H30 Cluster analysis; Secondary: 68T10 Pattern recognition

## 1 Introduction

Let the  $M$  columns of the matrix  $X \in \mathbb{R}^{N \times M}$  denote points in  $\mathbb{R}^N$  that we would like to cluster, using the  $K$ -means algorithm [4, 9] for example. The word cluster does not seem to have a unique mathematical meaning in the literature, but is used in a variety of situations for different purposes [5]. Loosely speaking, one would like to partition the  $M$  columns  $X_j$  of  $X$  into mutually exclusive subsets such that columns in the same subset are close to each other, while columns in different subsets

---

This material is based upon work supported by the National Science Foundation under Grant No. CCF-1450321.

---

✉ S. Chandrasekaran  
shiv@ece.ucsb.edu

<sup>1</sup> Department of Electrical and Computer Engineering, University of California, Santa Barbara, USA

are far apart from each other. Suppose we divide the columns into  $K$  subsets  $\mathcal{S}_k$  for  $0 \leq k < K$ . Then a possible mathematical problem that captures the intent is to pick subsets  $\mathcal{S}_k$  such that the sum of the squares of the intra-cluster separations,  $\sum_{0 \leq k < K} \sum_{X_i, X_j \in \mathcal{S}_k} \|X_i - X_j\|^2$ , is minimized, where  $\|\cdot\|$  denotes the Euclidean norm.

One reason for squaring the distance is to simplify the optimization algorithm when the Euclidean norm is used. However, for reasons of practicality and efficiency, the  $K$ -means algorithm uses a different formulation, which is equivalent in the case of squared Euclidean distance with weights. The  $\mathcal{S}_k$  are restricted to Voronoi cells. To each  $\mathcal{S}_k$  there is assigned a column  $Y_k$  such that  $x \in \mathcal{S}_k$  if  $k$  is the smallest integer for which  $\|x - Y_k\| = \min_{0 \leq l < K} \|x - Y_l\|$ . While this can be used with arbitrary norms, that is of no interest in this paper, and we will continue assuming that the norm is the standard Euclidean norm. In this formulation it is usually conventional to measure the goodness of a clustering via the expression,  $\sum_{0 \leq k < K} \sum_{X_j \in \mathcal{S}_k} \|X_j - Y_k\|^2$ . This considerably decreases the flop count of algorithms that try to minimize the above expression, as there are many fewer terms involved if  $K \ll M$ . There are many algorithms that directly or indirectly try to minimize the above expression over the  $K$  columns  $Y_j$ . However it is difficult to find the global minimum and the quality of the local minimum may not be good, though there does not necessarily seem to be agreement over this in the literature, as the precise local minima at which the algorithm stops depends on the starting point [13, 14, 18].

The aim of this paper is to consider a larger class of objective functions for choosing the partitioning of the columns  $X_j$ , in order to provide more flexibility in practice, while at the same time retaining the guaranteed descent feature of the standard  $K$ -means algorithm. In fact, the standard  $K$ -means algorithm will be a special case. However, we do not claim that the clusters our algorithm computes will be better in practice; this must be determined by the data and intended use of the clusters (we provide numerical experiments on some synthetic data sets in Sect. 4). Nor do we provide statistical justifications for our choice of objective functions, even though these would clearly be of great interest, as this would be non-trivial and would deviate from the main aim of the paper.

We propose two changes to the standard  $K$ -means objective function. The first is to bring in two levels of hierarchical clustering, and the second is to bring in explicit penalty terms for inter-cluster distances. To mimic the second level of hierarchical clustering we represent the subset  $\mathcal{S}_k$  as a union of Voronoi cells  $\mathcal{S}_{k;l}$ ; that is,  $\mathcal{S}_k = \cup_l \mathcal{S}_{k;l}$ . To penalize gaps between clusters we bring in terms of the form  $-\|Y_{k;l} - Y_{p;i}\|^2$ . Note the negative sign which encourages these distances to become larger during the optimization process. The rest of the paper works through the details of a  $K$ -means style algorithm with guaranteed descent. One notable aspect of the algorithm is the inclusion of a cheap combinatorial search phase that enables the algorithm to escape local minima with guaranteed descent.

## 1.1 Existing work

There exists a vast literature on different approaches to clustering, originating from seemingly disparate applications in chemistry, physics, biology, and medicine. Since the invention of the  $K$ -means algorithm, significant progress has been made in both the analysis and algorithmic approaches to the basic problem. For a comprehensive summary of early literature see [2, 5, 20].

Despite its prominent success, the  $K$ -means algorithm is not ideal for all clustering problems. When there truly are  $K$  clusters, and enough effort is expended, then, in some cases,  $K$ -means will converge quickly to the right solution [13, 19]. On the other hand, it is known that for ill-fated configurations  $K$ -means can take a long time to converge [18]. As such,  $K$ -means must be significantly tailored and tested for use in practical applications. Several improvements have been made to various aspects of the original algorithm proposed by Lloyd (1957) and Forgy (1965), including optimality [21, 22], efficiency [23, 24], heuristic model-order estimation and seeding [11, 25, 26], cluster-separation measures [27], and hierarchical representations [28, 29].

The objective of this paper is not to replace  $K$ -means with a better algorithm; rather, it is to present a new class of objective functions that incorporate some improvements with an associated fast algorithm. The work of Lindsten et al. [8, 12], for example, replaces the  $K$ -means objective function with a convex objective function with one continuous regularization parameter that replaces the discrete parameter  $K$ . This objective function bears some resemblance to ours. However, we note that their objective function has sums of norms, and not their squares, and so is more expensive to optimize. Furthermore, there are no negative terms in their objective function and hence no explicit penalty for inter-cluster distances. Finally every cluster is represented by a single center as in the classical  $K$ -means. Another modified  $K$ -means objective function is that of [10]. However, this does not include negative terms either.

Current hierarchical clustering algorithms are also asymptotically slower than  $K$ -means, but there has been work on making them faster, for example [1, 3]. Our algorithm leans heavily on the standard Euclidean norm, but other measures of similarity can be important in practice [17]. The multi-point representation of clusters is implicit in the work of Rose *et al.* on deterministic annealing [14], and is also common in applications of learning vector quantization [6]. Fuzzy clustering is another very popular technique and a large number of objective functions are known for which efficient local minima can be computed [15]. But none of these encompass the objective functions we discuss in this paper.

## 1.2 Notation

Let  $\mathbb{R}$  denote the set of reals,  $\mathbb{N}$  the set of non-negative integers, and  $\mathbb{N}_+$  the set of positive integers. Let  $\mathbb{N}_p = \{0, 1, \dots, p - 1\}$ , for  $p \in \mathbb{N}$ , with  $\mathbb{N}_0 = \{\}$ . Let  $\|\cdot\|$  denote the standard Euclidean 2-norm. Let  $\|\cdot\|_F$  denote the Frobenius norm. Let  $e$  denote the column vector of all ones; the dimension will be apparent from the context.

Breaking from custom, we will place row indices on the left. For example,  $iA_j$  will denote the  $(i, j)$ -th entry of the matrix  $A$ . We will also use  $A_j$  to denote the  $j$ -th column of  $A$ , while  $iA$  will denote the  $i$ -th row of  $A$ . We will use a double index notation for block matrices. So  $p;Ak;$  will denote the  $(p, k)$ -th block sub-matrix of  $A$ , and  $p;iA_k;j$  will denote the  $(i, j)$ -th entry of the block  $p;A_k$ . Frequently our row and column indices will start with 0 rather than 1.

Let  $N, L, K \in \mathbb{N}_+$ . Let  $Y \in \mathbb{R}^{N \times L}$ . Block partition the columns of  $Y$  into  $K$  blocks:

$$Y = (Y_0; Y_1; \dots Y_{K-1};).$$

Let  $\lambda \in \mathbb{N}_+^K$  for  $K \in \mathbb{N}_+$ , and let  $Y_{k;} \in \mathbb{R}^{N \times \lambda_k}$  for  $k \in \mathbb{N}_K$ ; that is,  $\lambda_k$  denotes the number of columns in  $Y_{k;}$  and  $\sum_{k \in \mathbb{N}_K} \lambda_k = L$ . Using  $Y$  and  $\lambda$ , partition  $\mathbb{R}^N$  into  $K$  mutually disjoint subsets  $\mathcal{S}_k$  according to the following membership rule:  $x \in \mathbb{R}^N$  is assigned to  $\mathcal{S}_k$  if  $k$  is the smallest integer for which

$$\min_{l \in \mathbb{N}_{\lambda_k}} \|x - Y_{k;l}\| = \min_{p \in \mathbb{N}_K} \min_{j \in \mathbb{N}_{\lambda_p}} \|x - Y_{p;j}\|.$$

Let  $\mathcal{S}_{k;l}$ , for  $l \in \mathbb{N}_{\lambda_k}$ , denote  $\lambda_k$  mutually disjoint subsets of  $\mathcal{S}_k$ . The membership rule for  $\mathcal{S}_{k;l}$  is as follows:  $x \in \mathcal{S}_k$  is assigned to  $\mathcal{S}_{k;l}$  if  $l$  is the smallest integer for which

$$\|x - Y_{k;l}\| = \min_{n \in \mathbb{N}_{\lambda_k}} \|x - Y_{k;n}\|.$$

We will call  $\mathcal{S}_{k;l}$  as a sub-cluster and  $\mathcal{S}_k$  as a cluster. Let  $X_{k;l}$  denote the sub-matrix of  $X$  that contains all the columns of  $X$  that lie in  $\mathcal{S}_{k;l}$ .

### 2 Problem

Let  $M, N \in \mathbb{N}_+$ . Let  $1 < L_1 \in \mathbb{N}_+$ . Let  $X \in \mathbb{R}^{N \times M}$  and  $\Omega \in \mathbb{R}^N$  be given. Let  $\alpha, \beta \geq 0$  and  $\varsigma, \gamma > 0$  be given. Let  $Y \in \mathbb{R}^{N \times L}$  for some  $0 < L \leq L_1$ . Let  $K \in \mathbb{N}_+$  and  $\lambda \in \mathbb{N}_+^K$  such that  $\sum_{k \in \mathbb{N}_K} \lambda_k = L$ . Let

$$F(Y, \lambda) = \sum_{k \in \mathbb{N}_K} \sum_{l \in \mathbb{N}_{\lambda_k}} \|X_{k;l} - Y_{k;l}e^T\|_F^2 + \alpha \sum_{k \in \mathbb{N}_K} \sum_{l < n \in \mathbb{N}_{\lambda_k}} \|Y_{k;l} - Y_{k;n}\|^2 + \frac{\beta}{\gamma} \sum_{k < p \in \mathbb{N}_K} \sum_{l \in \mathbb{N}_{\lambda_k}, j \in \mathbb{N}_{\lambda_p}} (1 - \gamma \|Y_{k;l} - Y_{p;j}\|^2) + \varsigma \|Y - \Omega e^T\|_F^2. \tag{1}$$

Given  $X, \Omega, \alpha, \beta, \varsigma, \gamma, L_1$ , find  $L, Y, K$  and  $\lambda$ , which solves the minimization problem

$$\min_{Y, \lambda} F(Y, \lambda), \quad \text{when } \beta < \frac{\varsigma}{2(L_1 - 1)}.$$

**Table 1** Inputs

$N$	Dimensionality of vectors to be clustered
$M$	Number of vectors to be clustered
$X$	$N \times M$ matrix of vectors to be clustered
$L_1$	Maximum number of sub-clusters allowed

**Table 2** Outputs

$K$	Number of clusters found
$L$	Total number of sub-clusters found
$Y$	$N \times K$ matrix of sub-cluster centers
$\lambda$	$K$ dimensional vector of number of sub-clusters in each cluster
$Y_k;$	$N \times \lambda_k$ matrix of sub-cluster centers of cluster $k$
$Y_{k;l}$	Center of sub-cluster $l$ in cluster $k$

**Table 3** Algorithm parameters

$\Omega$	Best chosen to be the mean of the columns of $X$
$\alpha$	Larger values forces sub-clusters of a single cluster to lie closer together
$\beta$	Larger values forces clusters apart
$\gamma$	Larger values increases minimum gap between clusters
$\varsigma$	Small number that prevents the $Y$ 's from drifting too far from $\Omega$

For clarity, we have summarized the notation in Tables 1, 2 and 3. Note that the term  $\|X_{k;l} - Y_{k;l}e^T\|_F^2$  is the usual penalty on the distance of a column of  $X$  from its assigned sub-cluster center. The term  $\|Y_{k;l} - Y_{k;n}\|^2$  is a penalty on the distance between the centers of two sub-clusters that belong to the same cluster. The term  $-\|Y_{k;l} - Y_{p;j}\|^2$  is a penalty on the distance between sub-cluster centers belonging to two different clusters. Note the negative sign as we want this term to be large in absolute value. This is also the term that makes the objective function indefinite. The term  $\|Y - \Omega e^T\|_F^2$  is a penalty on the distance of the sub-cluster centers from  $\Omega$  and is there to prevent the objective function from becoming unbounded from below.

The global optimum is hard to find, so we settle for a “local” minima, though the word “local” is dubious in a discrete setting. The bound on  $\beta$  is needed to ensure that  $F$  is bounded from below. We recommend choosing  $\gamma$  to be reasonably small so as to discourage the formation of empty sub-clusters (see Proposition 5). A good default choice for  $\Omega$  is the global mean  $\Omega = \frac{Xe}{M}$ . The role of  $\alpha$  is to encourage sub-clusters belonging to a single cluster to be close together, while the role of  $\beta$  and  $\gamma$  is to encourage clusters to be well-separated. The role of  $\varsigma$  is purely technical at this point; it keeps  $F$  bounded from below when some sub-clusters become empty.

There are several components to this problem and it is difficult to find a linear presentation. Assuming that the reader is familiar with the  $K$ -means algorithm, we begin with a rough outline of the algorithm and then present the details. Our goal is a guaranteed descent algorithm to a local minimum.

### 3 The Algorithm

The algorithm is a form of block coordinate descent, complicated by the presence of a combinatorial part. The algorithm proceeds in multiple stages. In each stage we guarantee that  $F$  is non-increasing.

1. Initialize  $Y$  (essentially randomly from columns of  $X$ ) with  $K = L_1$  (Sect. 3.1).
2. Assign columns of  $X$  by the nearest center rule (Sect. 3.2).
3. **Repeat:**
  - A. Compute  $C$ ,  $T$  and  $R$  (Sect. 3.3).
  - B. **For each** column  $Y_{k;l}$ :
    - (a) **If** sub-cluster  $\mathcal{S}_{k;l}$  is empty delete if descent is possible (Sect. 3.3.2).
    - (b) **Else** among the following choices, **pick** the **one** with maximum descent:
      - (i) Split off sub-cluster into its own cluster if descent is possible (Sect. 3.3.3).
      - (ii) Transfer sub-cluster to another cluster if descent is possible. (Sect. 3.3.4).
      - (iii) Swap with another sub-cluster if descent is possible (Sect. 3.3.5).
    - (c) Update  $\lambda$ ,  $T$ ,  $R$  and other variables as needed.
  - C. Freeze all partitions  $\mathcal{S}_{k;l}$  and move  $Y$  to the nearest critical point (Sect. 3.4).
  - D. **For each** column  $X_j$ :
    - (a) **If**  $L < L_1$  and if  $X_j = Y_{K;0}$  would lead to descent take this path (Sect. 3.3.1).
    - (b) **Else** assign to nearest  $Y_{k;l}$  (guarantee descent) (Sect. 3.2).
    - (c) **If**  $X_j$  changed membership, freeze all the partitions  $\mathcal{S}_{k;l}$  and modify  $Y$  to reach nearest local minimum (guarantee descent) (Sect. 3.4).
4. **Until** no (significant) descent

As mentioned earlier, the  $K$ -means objective can be attained by forcing all clusters to have only one sub-cluster, thereby skipping step 3B and effectively eliminating the first penalty term in the objective function  $F$ . We point out a key difference with what most people call Lloyd's [9] or Forgy's [4] version of the  $K$ -means algorithm: we update the centers every time  $X_j$  is re-assigned. This second version of  $K$ -means is known to be more efficient in the Euclidean case [7, 16]. Furthermore, it guarantees (modulo floating-point errors) that no empty clusters will be produced by  $K$ -means, which is a frequent problem in Forgy's version.

**Proposition 1** *The cost of one loop, steps 3B, 3C and 3D, is  $O(NML + L^2)$  flops. Each step of the loop is guaranteed not to increase the objective function.*

*Proof* Established in the propositions below. □

#### 3.1 Initializing $Y$

We choose the initial cluster centers  $Y$  by common randomization techniques. Let  $Q \in \mathbb{N}$  be a positive integer.

1. Choose  $Q$  columns of  $X$  randomly. Find one with the largest separation and keep as the first column of  $Y$ . That is,

$$Y_1 = \operatorname{argmax}_{X_i} \|X_i - X_j\| \quad \forall i, j \in \mathbb{N}_M$$

2.  $L_1 - 1$  times do the following:
  - (a) Choose  $Q$  columns randomly from  $X$ . Keep the one furthest from the current set of  $Y$  columns as the next column of  $Y$ . If the largest distance is zero, this step must be repeated until the largest distance becomes non-zero.

Choose  $K = L_1$ , so every column of  $Y$  corresponds to a cluster, and there are  $L_1$  initial clusters. As long as there are enough distinct columns in  $X$  this process will terminate in a finite number of iterations and cost  $O(Q^2N + QNL_1^2)$  flops. This is the costliest stage and to balance the cost we must pick  $Q$  such that  $Q \ll \min(\sqrt{MIL_1}, MI/L_1)$  where  $I$  is the number of iterations that the algorithm will run, which is of course not available a priori.

### 3.2 Assigning $X_j$

Column  $X_j$  is assigned to  $S_{k;l}$  following the standard membership rule described in Sect. 1.2. The cost of assigning one  $X_j$  is  $O(NL)$  flops. The objective function is guaranteed not to increase, and strict decrease is assured if  $X_j$  changes its membership. The first time we are guaranteed that each  $S_{k;l} = S_k$  will be non-empty. The remaining times this guarantee is not available. The total cost of assigning all columns of  $X$  is  $O(NML)$  flops.

### 3.3 Re-arranging sub-clusters

The goal here is to figure out to quickly “re-arrange” the sub-clusters such that  $F$  decreases. A symmetric two-dimensional array  $C \in \mathbb{R}^{L \times L}$  will be used to hold the distances between the columns of  $Y$ . Let

$$\begin{aligned} {}_{k;l}C_{k;l} &= \|Y_{k;l} - \Omega\|^2, \quad k \in \mathbb{N}_K, \quad l \in \mathbb{N}_{\lambda_k}, \\ {}_{k;l}C_{k;n} &= \|Y_{k;l} - Y_{k;n}\|^2, \quad l \neq n \in \mathbb{N}_{\lambda_k}, \\ {}_{k;l}C_{p;i} &= \|Y_{k;l} - Y_{p;i}\|^2, \quad k \neq p \in \mathbb{N}_K, \quad l \in \mathbb{N}_{\lambda_k}, \quad i \in \mathbb{N}_{\lambda_p}. \end{aligned}$$

The two-dimensional array  $T \in \mathbb{R}^{L \times K}$  will be used to hold the distances between columns of  $Y$  and sub-clusters, while the one-dimensional array  $R \in \mathbb{R}^L$  will hold the distances from columns of  $Y$  to all sub-clusters that it is not a member of. Let

$$\begin{aligned} {}_{k;l}T_k &= \sum_{l \neq n \in \mathbb{N}_{\lambda_k}} {}_{k;l}C_{k;n}, \quad k \in \mathbb{N}_K, \quad l \in \mathbb{N}_{\lambda_k}, \\ {}_{k;l}T_p &= \sum_{i \in \mathbb{N}_{\lambda_p}} {}_{k;l}C_{p;i}, \quad k \neq p \in \mathbb{N}_K, \quad l \in \mathbb{N}_{\lambda_k}, \\ R_{k;l} &= \sum_{k \neq p \in \mathbb{N}_K} {}_{k;l}T_p, \quad k \in \mathbb{N}_K, \quad l \in \mathbb{N}_{\lambda_k}. \end{aligned}$$

**Proposition 2**  $C, T$  and  $R$  can be computed in  $O(NL^2)$  flops.

*Proof* It is easy to see that  $C$  can be computed in  $O(NL^2)$  flops.

${}_{k;l}T_k$  can be computed in  $O(\lambda_k)$  flops. With some care  ${}_{k;l}T_k$  can be computed in  $O(\lambda_k)$  flops too.  ${}_{k;l}T_p$  can be computed in  $O(\lambda_p)$  flops, and  ${}_{k;l}T_p$  can be computed in  $O(\lambda_p\lambda_k)$  flops. Hence  $T$  can be computed in  $O(L^2)$  flops.

$R_{k;l}$  can be computed in  $O(K)$  flops, and  $R_{k;l}$  can be computed in  $O(\lambda_k K)$  flops. Therefore  $R$  can be computed in  $O(KL)$  flops, or, more generously, in  $O(L^2)$  flops. □

Based on  $C, T$  and  $R$ , we re-arrange the sub-clusters, changing the membership of at most two columns of  $Y$  at a time, while still ensuring descent of  $F$ . If the sub-clusters are re-arranged then  $C, T$  and  $R$  must be updated efficiently by only re-computing numbers that have changed. The details are an on obvious calculation and are not presented in this paper.

In this section alone we will let  $F_1$  denote the value of  $F$  before the intended operation, and let  $F_2$  denote the value of  $F$  after the intended operation. The operation will cause a strict decrease in the value of  $F$ , if  $F_1 - F_2 > 0$ . There are four possible operations we consider for each column  $Y_{k;l}$ , and there is a fifth one for introducing a new  $Y_{k;l}$ .

Note that we do not entertain operations that look at three columns of  $Y$  at the same time, since the step will become  $L$  times slower. However, in some situations it might be worthwhile to do so, especially if  $L \sim N$ .

### 3.3.1 Assigning $X_j$ its own cluster

If  $L < L_1$  there is room to create new clusters.

**Proposition 3** If  $L < L_1$  and  $X_j \in S_{k;l}$ , then introducing  $Y_{K;0} = X_j$  will result in

$$F_1 - F_2 = \|X_j - Y_{k;l}\|^2 + \beta \left( \|X_j e^T - Y\|_F^2 - \frac{L}{\gamma} \right) - \varsigma \|X_j - \Omega\|^2.$$

*Proof* From Eq. (1) we can compute

$$F_1 - F_2 = \|X_j - Y_{k;l}\|^2 - \varsigma \|X_j - \Omega\|^2 - \frac{\beta}{\gamma} \left( L - \gamma \|X_j e^T - Y\|_F^2 \right).$$

□

This also gives one way to interpret  $\gamma$ , and one of the effects of  $\varsigma$ — it discourages the creation of clusters away from  $\Omega$ . It also shows one of the significant differences with  $K$ -means, where new clusters can be easily introduced, and the number of clusters must be controlled explicitly. In our algorithm the constants indirectly influence the number of clusters and should produce a smoother way to tune the algorithm in practice. Note that  $Y_{k;l}$  occurs twice in the above expression.

### 3.3.2 Deleting an empty sub-cluster

It possible that after the columns of  $X$  have been re-assigned to the columns of  $Y$ , some subset  $\mathcal{S}_{k;l}$  associated with  $Y_{k;l}$  might be empty. In this case we give priority to deleting this sub-cluster if possible and decrement  $L$  by 1 if we succeed.

**Proposition 4** *If  $\mathcal{S}_{k;l} = \{\}$  then deleting  $Y_{k;l}$  will result in*

$$F_1 - F_2 = \alpha_{k;l}T_k + \varsigma_{k;l}C_{k;l} + \frac{\beta}{\gamma}(L - \lambda_k) - \beta R_{k;l}. \tag{2}$$

Once  $C, T$  and  $R$ , are available, this can be computed in  $O(1)$  flops.

*Proof* From Eq. (6) we can compute

$$F_1 - F_2 = \alpha_{k;l}T_k + \frac{\beta}{\gamma}(L - \lambda_k - \gamma R_{k;l}) + \varsigma \leq k; lCk; l.$$

□

**Proposition 5** *If*

$$\gamma \leq \frac{\varsigma - 2\beta(L_1 - 1)}{2\left(\|X\|_2\sqrt{M} + \varsigma\|\Omega\|_2\sqrt{L_1}\right)}, \tag{3}$$

*then expression (2) is always non-negative.*

*Proof* Follows from Proposition 14. □

Thus with sufficiently small choice of  $\gamma$  the algorithm will always delete empty sub-clusters. We do not recommend setting  $\gamma$  this small as the upper bound is extremely loose and would not enforce large gaps between clusters. If  $Y_{k;l}$  is deleted then we need to update  $C, T$  and  $R$  efficiently the details of which are skipped in this paper.

### 3.3.3 Splitting off a sub-cluster

We now develop a criteria to check if  $Y_{k;l}$  should be split off into its own cluster in case  $\lambda_k > 1$ , and if  $K$  should be increased by one.

**Proposition 6** *If  $Y_{k;l}$  is split off into its own cluster*

$$F_1 - F_2 = (\alpha + \beta)_{k;l}T_k - \frac{\beta}{\gamma}(\lambda_k - 1).$$

*This can be computed in  $O(1)$  flops once  $T$  and  $\lambda$  are available.*

*Proof* We can compute from Eq. (6)

$$\begin{aligned}
 F_1 - F_2 &= \alpha_{k;l}T_k + \frac{\beta}{\gamma}(L - \lambda_k - \gamma R_{k;l}) + \varsigma_{k;l}C_{k;l} \\
 &\quad - \frac{\beta}{\gamma}(L - \lambda_k - \gamma R_{k;l}) - \varsigma_{k;l}C_{k;l} - \frac{\beta}{\gamma}(\lambda_k - 1 - \gamma_{k;l}T_k) \\
 &= (\alpha + \beta)_{k;l}T_k - \frac{\beta}{\gamma}(\lambda_k - 1).
 \end{aligned}$$

□

We note that large values for  $\gamma$  will encourage sub-clusters to split off. This is another reason to keep  $\gamma$  reasonably small. This also gives a good thumb rule for tuning  $\gamma$  on synthetic data sets. If  $Y_{k;l}$  is split off into its own cluster then  $C$ ,  $T$  and  $R$  have to be updated efficiently.

### 3.3.4 Transferring a sub-cluster to another cluster

**Proposition 7** *If  $Y_{k;l}$  is transferred from cluster  $k$  to cluster  $p$  then*

$$F_1 - F_2 = (\alpha + \beta)_{(k;l)T_k - (k;l)T_p} - \frac{\beta}{\gamma}(\lambda_k - \lambda_p - 1),$$

and this can be computed in  $O(1)$  flops once  $T$  and  $\lambda$  are available.

*Proof* We compute from Eq. (6)

$$\begin{aligned}
 F_1 - F_2 &= \alpha_{k;l}T_k + \varsigma_{k;l}C_{k;l} + \frac{\beta}{\gamma}(\lambda_p - \gamma_{k;l}T_p) + \frac{\beta}{\gamma}(L - \lambda_p - \lambda_k - \gamma(R_{k;l} -_{k;l}T_p)) \\
 &\quad - \alpha_{k;l}T_p - \varsigma_{k;l}C_{k;l} - \frac{\beta}{\gamma}(\lambda_k - 1 - \gamma_{k;l}T_k) \\
 &\quad - \frac{\beta}{\gamma}(L - \lambda_p - \lambda_k - \gamma(R_{k;l} -_{k;l}T_p)) \\
 &= (\alpha + \beta)_{k;l}T_k - (\alpha + \beta)_{k;l}T_p + \frac{\beta}{\gamma}(\lambda_p - \lambda_k + 1).
 \end{aligned}$$

□

If  $Y_{k;l}$  is transferred to cluster  $p$ , then  $T$  and  $R$  have to be updated efficiently. Note that if  $\gamma$  is not sufficiently large, huge clusters will gobble up small clusters.

### 3.3.5 Swapping two sub-clusters

**Proposition 8** *If  $Y_{k;l}$  is swapped with  $Y_{p;i}$  then*

$$F_1 - F_2 = (\alpha + \beta) (_{k;l}T_k + _{p;i}T_p - _{k;l}T_p - _{p;i}T_k + 2_{k;l}C_{p;i}).$$

*Proof* From Eq. (6) we can compute

$$\begin{aligned}
 F_1 - F_2 &= \alpha_{k;l}T_k + \frac{\beta}{\gamma}(\lambda_p - \gamma_{k;l}T_p) + \alpha_{p;i}T_p + \frac{\beta}{\gamma}(\lambda_k - \gamma_{p;i}T_k) \\
 &\quad - \alpha_{(p;i)T_k - p;i}C_{k;l}) - \alpha_{(k;l)T_p - k;l}C_{p;i}) - \frac{\beta}{\gamma}(\lambda_p - \gamma_{p;i}T_p \\
 &\quad - \gamma_{p;i}C_{k;l}) - \frac{\beta}{\gamma}(\lambda_k - \gamma_{k;l}T_k - \gamma_{k;l}C_{p;i}) \\
 &= (\alpha + \beta)_{k;l}T_k + (\alpha + \beta)_{p;i}T_p - (\alpha + \beta)_{k;l}T_p - (\alpha + \beta)_{p;i}T_k \\
 &\quad +_{p;i}C_{k;l}(\alpha + \beta) +_{k;l}C_{p;i}(\alpha + \beta).
 \end{aligned}$$

□

If the swap is carried out  $T$  and  $R$  must be updated.

### 3.4 Descending $Y$

Suppose we freeze the membership of the columns of  $X$  in the subsets  $\mathcal{S}_{k;l}$ . What is the optimal choice for  $Y$ ? In the  $K$ -means case the optimal choice is clearly the mean of each cluster. In our case the answer is only a little more complicated. Define the matrix  $A \in \mathbb{R}^{L \times L}$  as follows:

$$\begin{aligned}
_{k;l}A_{k;l} &= |\mathcal{S}_{k;l}| + \alpha(\lambda_k - 1) + \zeta - \beta(L - \lambda_k), \\
_{k;l}A_{k;n} &= -\alpha, \quad l \neq n \in \mathbb{N}_{\lambda_k}, \\
_{k;l}A_{p;i} &= +\beta, \quad k \neq p \in \mathbb{N}_K, \quad i \in \mathbb{N}_{\lambda_p},
 \end{aligned} \tag{4}$$

where  $|\mathcal{S}|$  denotes the cardinality of the set  $\mathcal{S}$ . Define the matrix  $W \in \mathbb{R}^{N \times L}$  as follows:  $W_{k;l} = X_{k;l}e$ . We assume that  $W_{k;l} = 0$  if  $\mathcal{S}_{k;l} = \{\}$ .

**Proposition 9** *For a fixed set of  $\mathcal{S}_{k;l}$  and a fixed  $\lambda$ , there is exactly one minimum point:*

$$Y = (W + \zeta \Omega e^T) A^{-1}.$$

*Proof* See proof of Proposition 15 and the discussion leading up to it. □

**Proposition 10** *For a fixed set of  $\mathcal{S}_{k;l}$  and a fixed  $\lambda$ , the unique critical point  $Y$  can be computed in  $O(NL)$  flops.*

*Proof* See Proposition 21 and the argument leading to it. □

Note that this cost is comparable to the cost of assigning one column  $X_j$  to its optimal sub-cluster. So re-computing  $Y$  for every such assignment only affects the constant in the flop count. However, the reason for doing so, is similar to that of the non-standard  $K$ -means algorithm: it reduces the chance of producing empty sub-clusters.

### 4 Numerical experiments

Tests were carried out on a set of random synthetic data consisting of square clusters (with no intentional sub-clusters). The data was generated in this manner to ensure that neither K-means nor our method would have an unfair advantage.

Let  $M_1 \in \mathbb{N}_+$ . Let  $P_{k_i} \in \mathbb{R}^{N \times M_1}$  be a random matrix with entries chosen uniformly from  $[-1, 1]$ . Let  $S \in \mathbb{R}^{N \times K}$  be a random matrix with entries chosen uniformly from  $[-1, 1]$ . Let  $D \in \mathbb{R}^{K \times K}$  be a diagonal matrix with entries chosen uniformly from  $[-\nu, \nu]$  for  $\nu \in \mathbb{R}$  and fixed. Let  $X_{k_i} = P_{k_i} + D_{k_i} S_{k_i} e^T$ . The assumption is that the columns of  $X_{k_i}$  belong to a single cluster.

Let  $U_{k_i}$  represent any other clustering of the columns of  $X$  into  $K_1$  clusters. Let  $k \mathfrak{C}_l$  denote the number of columns of  $X_l$  that are in  $U_{k_i}$ . Let

$$\mathfrak{R}(l) = \operatorname{argmax}_{k \in \mathbb{N}_{K_1}} k \mathfrak{C}_l,$$

and

$$\mathfrak{L}(k) = \operatorname{argmax}_{l \in \mathbb{N}_K} k \mathfrak{C}_l.$$

The true cluster  $X_{l_i}$  is largely in cluster  $U_{\mathfrak{R}(l)}$ , and cluster  $U_l$  largely consists of elements of true cluster  $X_{\mathfrak{L}(k)}$ . Let

$$\mathfrak{S}(U) = \sum_{l \in \mathbb{N}_K} \sum_{k \in \mathbb{N}_{K_1}, k \neq \mathfrak{R}(l)} k \mathfrak{C}_l + \sum_{k \in \mathbb{N}_{K_1}} \sum_{l \in \mathbb{N}_K, l \neq \mathfrak{L}(k)} k \mathfrak{C}_l.$$

( $\mathfrak{S}(U)$  is the sum of false positives and negatives when those concepts make sense.) We can simplify this

$$\begin{aligned} \mathfrak{S}(U) &= \sum_{l \in \mathbb{N}_K} (M_1 - \mathfrak{R}(l) \mathfrak{C}_l) + \sum_{k \in \mathbb{N}_{K_1}} (|U_{k_i}| - k \mathfrak{C}_{\mathfrak{L}(k)}) \\ &= 2M - \sum_{l \in \mathbb{N}_K} \mathfrak{R}(l) \mathfrak{C}_l - \sum_{k \in \mathbb{N}_{K_1}} k \mathfrak{C}_{\mathfrak{L}(k)}. \end{aligned}$$

Using the nearest center rule we can partition the columns of  $X$  into  $K$  clusters  $\Xi_{k_i}$  using the  $K$  columns of  $SD$ . Note that these may not be the same as  $X_{k_i}$ . Let  $Y$  and  $\Lambda$  be computed by the new method IMP, and let the resulting clustering of the columns of  $X$  into  $K_1$  clusters be denoted by  $Z_{k_i}$ . We will measure the goodness of  $Y$  and  $\Lambda$  by the number

$$\text{Score IMP} = \mathfrak{S}(\Xi) - \mathfrak{S}(Z).$$

This number lies in the range  $[-2M, 2M]$  and bigger numbers are taken to indicate that the clustering  $Z$  was good in some sense. Note that one advantage of this measure is that it does not benefit lumping all of  $X$  into a single cluster, or, breaking it all up into  $M$  clusters, since every clustering is compared to the putative right clustering  $X_{k_i}$ .

One disadvantage is that sometimes  $X_{k_i}$  is not the right clustering and our method can be penalized for finding it. Note that a score of 0 should be considered as excellent for this data set.

We will also compare against a  $K$ -means algorithm. Our implementation uses the same initialization routine as IMP except with a much larger value of  $Q$ . It also updates the cluster centers every time a column of  $X$  is re-assigned. We did this so as to avoid empty clusters. The  $K$ -means algorithm was provided with the correct value of  $K$  and ran it until it reached a local minimum. In fact we did this for both algorithms. IMP was provided with a starting value of  $L_1 = 2K$  and left to work out the true number of clusters. Both algorithms were run until they reached their local minimum and no attempt was made at early termination.

We chose some of the parameters as follows:

$$\begin{aligned}
 L_1 &= 2K, \\
 M &= \begin{cases} KM_1, & \text{no outliers,} \\ (K+1)M_1, & \text{with } M_1 \text{ outliers,} \end{cases} \\
 \varsigma &= \frac{M_1}{2000}, \\
 \beta &= \frac{\varsigma}{2.00001(L_1 - 1)}, \\
 \alpha &= 2(L_1 - 1)\beta, \\
 Q \text{ (for } K\text{-means)} &= K, \\
 Q \text{ (for fIMP)} &= \begin{cases} 30, & \text{if } K = 100, \\ 40, & \text{if } K = 200, \end{cases} \\
 \nu &= 10K^{1/N} \log_2(K), \\
 \text{Number of trials} &= \begin{cases} 100, & \text{if } K = 100, \\ 50, & \text{if } K = 200. \end{cases}
 \end{aligned}$$

The experimental results for  $K = 100$  are summarized in Table 4 and for  $K = 200$  in Table 5. In these and subsequent tables we use the following notation for the column labels. The column  $\text{Losses}_1$  reports the number of times the IMP score was strictly negative out of 100 trials. The column “IMP score” reports the average score for IMP across 100 trials. The column  $\text{Losses}_2$  reports the number of times the score for IMP was strictly worse than the score for  $K$ -means out of all trials. The column “ $K$ -means score” reports the average score for  $K$ -means. The column “Time IMP” reports the average running time for IMP in seconds. The column “Time  $K$ -means” reports the average running time for  $K$ -means in seconds.

For this specific synthetic data set and initialization strategy we conjecture that IMP takes  $O(NMK^2)$  flops to find a local minima while  $K$ -means takes  $O(NMK)$  flops. We conjecture that on average IMP comes close to the global minimum while  $K$ -means is off by about one cluster. It is crucial to note that this synthetic data set has no outliers, the clusters tend to collide more frequently near the origin, and the clusters are convex (cubical) in shape. So these conjectures are only in this very restrictive setting.

**Table 4** Experimental results for  $K = 100$  over 100 trials

$N$	$M_1$	$\gamma \times 10^4$	Losses <sub>1</sub>	IMP score	Losses <sub>2</sub>	$K$ -means score	Time IMP	Time $K$ -means
7	30	4	42	-6	8	-31	0.60	0.04
14	30	4	53	-6	14	-28	1.08	0.07
28	60	4	47	-8	11	-58	4.03	0.26
56	120	3	26	-15	1	-129	17.32	1.16
112	240	3	23	-14	6	-237	80.81	5.35

**Table 5** Experimental results for  $K = 200$  over 50 trials

$N$	$M_1$	$\gamma \times 10^4$	Losses <sub>1</sub>	IMP score	Losses <sub>2</sub>	$K$ -means score	Time IMP	Time $K$ -means
7	30	2.5	36	-8	9	-39	4.60	0.23
14	30	2	40	-14	9	-48	8.12	0.39
28	60	2	33	-16	4	-83	29.60	1.23
56	120	2	33	-37	8	-158	138.91	5.60
112	240	0.8	16	-73	3	-296	665.94	23.98

Note that compared to Table 4 both  $M$  and  $K$  are doubled in corresponding rows

**Table 6** Experimental results for  $K = 100$  with  $M_1$  outliers over 100 trials

$N$	$M_1$	$\gamma \times 10^4$	Losses <sub>1</sub>	IMP score	Losses <sub>2</sub>	$K$ -means score	Time IMP	Time $K$ -means
7	30	4	33	-6	12	-24	0.45	0.06
14	30	4	54	-9	12	-27	0.80	0.10
28	60	4	32	-8	0	-86	2.33	0.57
56	120	3	21	-15	0	-400	9.71	4.00
112	240	3	14	-26	0	-1428	46.36	32.22

**Table 7** Experimental results for  $K = 200$  with  $M_1$  outliers over 50 trials

$N$	$M_1$	$\gamma \times 10^4$	Losses <sub>1</sub>	IMP score	Losses <sub>2</sub>	$K$ -means score	Time IMP	Time $K$ -means
7	30	2	31	-14	4	-44	3.98	0.29
14	30	2	39	-10	11	-33	6.99	0.45
28	60	2	34	-21	2	-90	22.13	2.00
56	120	2	25	-27	0	-337	96.93	11.89
112	240	1	19	-71	0	-1602	500.21	84.31

Table 4 both  $M$  and  $K$  are doubled in corresponding rows

The next set of experiments was essentially a repeat of the previous set with outliers thrown in. In particular for every run we added  $M_1$  points distributed randomly in  $[-\nu, \nu]^N$ . The scoring however was restricted to the non-outliers and there was no penalty for empty clusters. To enable  $K$ -means to do well in this situation we seeded it with  $K + M_1$  centers. For IMP we chose  $L_1 = 2K$  as usual. The results for  $K = 100$  are shown in Table 6 and for  $K = 200$  shown in Table 7.

## 5 Conclusion

We have presented a single family of new objective functions for clustering. The advantage of this family is that it retains the efficient time complexity of  $K$ -means while allowing a different set of local minima that can be tuned via a few parameters. However, there is a much larger family of objective functions that can be explored. For example, we could consider other power laws on the distance, and we can allow many more constants. Our objective function can be viewed as an average linkage 2-level hierarchical clustering with gap penalties scheme. The linkage can be viewed as a complete graph with cluster centers as vertices. A very useful model would be to replace this graph with a spanning tree that is chosen dynamically to allow non-spherical clusters and decrease the objective function (single linkage with gap penalties). However the details for the corresponding fast solver becomes more complicated, so this will be addressed in a separate paper.

Our objective function corresponds to a spring–mass model, where some of the springs can be viewed either as having a negative spring constant, or, as being wrapped around through the point at  $\infty$ . Based on this physical model, one can see that we can consistently develop other objective functions, for example using spring–mass–charge models (the exponents will no longer just be  $+2$ ). What these models will lose is the unique local minima for a fixed partition, but fast gradient descent will still be possible. The details will be presented elsewhere.

It also would be nice to develop simple statistical models that can guide the user in the choice of the objective functions.

Our algorithm requires  $O(L^2)$  working space memory. One can implement an algorithm that requires less working space memory but more flops. Our algorithm has an  $O(L^2)$  combinatorial part in each iteration. This makes it non-scalable with respect to the number of clusters. We can make it  $O(L)$  by only considering  $O(1)$  random cluster centers during the combinatorial phase. The details will be presented elsewhere.

Last, but not least, a C implementation of the algorithm is available from our website <http://scg.ece.ucsb.edu>.

**Acknowledgements** We would like to thank Jiyun Byun, Ken Rose and Ken Sullivan for useful discussions.

## Appendix

### Hessian of $F$

We begin with the calculation and simplification of the Hessian of  $F$ . For arbitrary matrices  $B$  and  $C$

$$B \otimes C = \begin{pmatrix} {}_1B_1 C & {}_1B_2 C & \cdots \\ {}_2B_1 C & {}_2B_2 C & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad \text{and} \quad \text{vec}(B) = \begin{pmatrix} B_1 \\ B_2 \\ \vdots \end{pmatrix}.$$

For square matrices  $B$  we define  $\text{tr}(B) = \sum_{i=1}^n B_i$ . Note that for a matrix  $B$  with  $n$  columns

$$\|B - ye^T\|_F^2 = \|B\|_F^2 - 2y^T B e + \|y\|^2 n. \tag{5}$$

Let  $Y_{-k;l}$  denote the sub-matrix of  $Y_k$ ; obtained by dropping  $Y_{k;l}$  from  $Y_k$ . Let  $Y_{-k}$ ; denote the sub-matrix of  $Y$  obtained by dropping the sub-matrix  $Y_k$ ; from  $Y$ . All the terms in  $F$  that depend on the single column  $Y_{k;l}$  are

$$\begin{aligned} F(Y, \lambda) = & \|Y_{k;l}e^T - X_{k;l}\|_F^2 + \alpha \|Y_{k;l}e^T - Y_{-k;l}\|_F^2 \\ & + \frac{\beta}{\gamma} \left( L - \lambda_k - \gamma \|Y_{k;l}e^T - Y_{-k}\|_F^2 \right) \\ & + \varsigma \|Y_{k;l} - \Omega\|^2 + \text{terms independent of } Y_{k;l}. \end{aligned} \tag{6}$$

We can use identity (5), expand and gather terms to obtain

$$\begin{aligned} F(Y, \lambda) = & \|X_{k;l}\|_F^2 + \alpha \|Y_{-k;l}\|_F^2 + \frac{\beta}{\gamma} \left( L - \lambda_k - \gamma \|Y_{-k}\|_F^2 \right) + \varsigma \|\Omega\|^2 \\ & - 2Y_{k;l}^T (X_{k;l}e + \alpha Y_{-k;l}e + \varsigma \Omega - \beta Y_{-k}e) + \|Y_{k;l}\|^2 (|\mathcal{S}_{k;l}| \\ & + \alpha(\lambda_k - 1) + \varsigma - \beta(L - \lambda_k)) + \text{terms independent of } Y_{k;l}. \end{aligned} \tag{7}$$

From Eq. (7) we can compute the gradient:

$$\frac{1}{2} \frac{\partial F(Y, \lambda)}{\partial Y_{k;l}} = Y_{k;l} (|\mathcal{S}_{k;l}| + \alpha(\lambda_k - 1) + \varsigma - \beta(L - \lambda_k)) - (X_{k;l}e + \alpha Y_{-k;l}e + \varsigma \Omega - \beta Y_{-k}e). \tag{8}$$

Note that this is the  $(k; l)$ -th block component of the gradient viewed as a column vector. It is useful to re-write this in a simpler form. First we observe that the matrix  $A$  defined in Eq. (4) is symmetric;  $A = A^T$ . Now we can write the gradient of  $F$  as

$$\frac{1}{2} \frac{\partial F(Y, \lambda)}{\partial Y} = YA - (W + \varsigma \Omega e^T), \tag{9}$$

where the gradient is now written as a matrix for convenience. In standard column form we can write

$$\frac{1}{2} \frac{\partial F(Y, \lambda)}{\partial Y} = (A \otimes I) \text{vec}(Y) - \text{vec}(W) - \varsigma e \otimes \Omega, \tag{10}$$

where we used the fact that  $A$  is symmetric.

**Proposition 11** *A is strictly diagonally dominant and positive definite and*

$$\|A^{-1}\|_2 < \frac{1}{\varsigma - 2\beta(L - 1)},$$

when  $L > 1$ .

*Proof* We claim the diagonal entries are positive since

$$|\mathcal{S}_{k;l}| + \alpha(\lambda_k - 1) + \varsigma - \beta(L - \lambda_k) \geq \varsigma - (L - 1)\beta > 0,$$

where we used the assumptions that  $\lambda_k \geq 1$ , and

$$\beta < \frac{\varsigma}{2(L_1 - 1)} \leq \frac{\varsigma}{2(L - 1)}.$$

The sum of the absolute values of the entries in row  $(k; l)$  is given by  $\alpha(\lambda_k - 1) + \beta(L - \lambda_k)$ . We claim that this sum is strictly smaller than the corresponding diagonal term since

$$|\mathcal{S}_{k;l}| + \alpha(\lambda_k - 1) + \varsigma - \beta(L - \lambda_k) - \alpha(\lambda_k - 1) - \beta(L - \lambda_k) \geq \varsigma - 2\beta(L - 1) > 0.$$

Applying Gerschgorin's theorem we get  $\lambda_{\min}(A) \geq \varsigma - 2\beta(L - 1) > 0$ , from which we obtain the desired upper bound on the 2-norm of  $A^{-1}$ .  $\square$

### Proposition 12

$$\|W\|_2 \leq \|X\|_2 \sqrt{M}.$$

*Proof* From  $W_{k;l} = X_{k;l} e$  we obtain  $\|W_{k;l}\|_2 \leq \|X\|_2 \sqrt{|\mathcal{S}_{k;l}|}$ . Therefore  $\|W\|_2 \leq \|W\|_F \leq \|X\|_2 \sqrt{M}$ .  $\square$

### Proposition 13 All critical points of $F$ are uniformly bounded.

*Proof*  $L > 1$  is the non-trivial case. From Proposition 11, it follows that the critical points that are solutions of the equation

$$\frac{\partial F(Y, \lambda)}{\partial Y} = YA - (W + \varsigma\Omega e^T) = 0,$$

satisfy the bound

$$\|Y\| = \left\| (W + \varsigma\Omega e^T) A^{-1} \right\| \leq \|W + \varsigma\Omega e^T\| \|A^{-1}\| \leq \frac{\|W\| + \varsigma\|\Omega\|\|e\|}{\varsigma - 2\beta(L - 1)} < \infty,$$

for any sub-multiplicative norm. Since  $F$  is differentiable and bounded from below, there are no other critical points to consider. Using Proposition 12 we also have the explicit uniform upper bound

$$\|Y\|_2 \leq \frac{\|X\|_2 \sqrt{M} + \varsigma\|\Omega\|_2 \sqrt{L}}{\varsigma - 2\beta(L - 1)} \leq \frac{\|X\|_2 \sqrt{M} + \varsigma\|\Omega\|_2 \sqrt{L_1}}{\varsigma - 2\beta(L_1 - 1)}.$$

$\square$

**Proposition 14**

$$R_{k;l} \leq \frac{2L_1 \left( \|X\|_2 \sqrt{M} + \varsigma \|\Omega\|_2 \sqrt{L_1} \right)}{\varsigma - 2\beta(L_1 - 1)}.$$

*Proof* Follows from the previous proposition and the easily established upper bound  $R_{k;l} \leq 2L_1 \|Y\|_2$ . □

From Eq. (8) we can compute the Hessian of  $F$ , denoted as  $2H$ :

$$\begin{aligned} {}_{k;l}H_{k;l} &= \frac{1}{2} \frac{\partial^2 F(Y, \lambda)}{\partial^2 Y_{k;l}} = (|S_{k;l}| + \alpha(\lambda_k - 1) + \varsigma - \beta(L - \lambda_k))I, \\ {}_{k;l}H_{k;n} &= \frac{1}{2} \frac{\partial^2 F(Y, \lambda)}{\partial Y_{k;l} \partial Y_{k;n}} = -\alpha I, \\ {}_{k;l}H_{p;i} &= \frac{1}{2} \frac{\partial^2 F(Y, \lambda)}{\partial Y_{k;l} \partial Y_{p;i}} = \beta I. \end{aligned}$$

We can represent the Hessian in matrix form as  $H = A \otimes I$ , which also follows from Eq. (10).

**Proposition 15** *All critical points of  $F$  are of the form  $Y = (W + \varsigma \Omega e^T)A^{-1}$  and correspond to local minima of  $F$ .*

*Proof* Follows from the positive-definiteness of the Hessian  $H$ . □

Note that the formula for the critical point is a bit deceptive in appearance. For example, there is more than one critical point, since the choice of  $\lambda$  and  $S_{k;l}$  determine  $W$  and  $A$ .

**Rapid application of  $A^{-1}$**

We will depend on the Sherman–Morrison–Woodbury (SMW) formula

$$(I + UV^T)^{-1} = \left( I - U(I + V^T U)^{-1} V^T \right). \tag{11}$$

In this section let  $D$  denote the diagonal matrix  ${}_{k;l}D_{k;l} = |S_{k;l}| + \alpha\lambda_k + \varsigma - \beta(L - \lambda_k)$ . All  ${}_{k;l}D_{k;l}$  can be computed in  $O(L)$  flops once  $|S_{k;l}|$  is known. Note that  ${}_{k;l}D_{k;l} = {}_{k;l}A_{k;l} + \alpha \geq 0$ , since we have assumed that  $\alpha \geq 0$ . Let  $B$  denote the block-diagonal matrix  ${}_k B_k = (\alpha + \beta)ee^T$ , for  $k \in \mathbb{N}_K$ , where the size of each block is chosen such that  $A = D - B + \beta ee^T$ . Taking the cue from SMW, we first compute  $(D - B)^{-1}$  noting that we just have to invert the  $K$  diagonal blocks

$$\begin{aligned} {}_k D_k; - {}_k B_k &= {}_k D_k; - (\alpha + \beta)ee^T \\ &= {}_k D_k;^{1/2} \left( I - (\alpha + \beta) {}_k D_k;^{-1/2} ee^T {}_k D_k;^{-1/2} \right) {}_k D_k;^{1/2}. \end{aligned}$$

Let  $\tau_k = e^T {}_{k;l}D_{k;l}^{-1}e = \sum_{l \in \mathbb{N}_{\lambda_k}} {}_{k;l}D_{k;l}^{-1}$ , where  $\tau \in \mathbb{R}^K$  can be computed in  $O(L)$  flops if  ${}_{k;l}D_{k;l}$  is available.

**Proposition 16**

$$({}_{k;l}D_{k;l} - {}_{k;l}B_{k;l})^{-1} = {}_{k;l}D_{k;l}^{-1} + \frac{\alpha + \beta}{1 - (\alpha + \beta)\tau_k} {}_{k;l}D_{k;l}^{-1} e e^T {}_{k;l}D_{k;l}^{-1}.$$

*Proof* By Eq. (11). □

**Proposition 17**

$$\sigma = e^T (D - B)^{-1}e = \sum_{k \in \mathbb{N}_K} \frac{\tau_k}{1 - (\alpha + \beta)\tau_k}.$$

*Proof* By direct calculation. □

**Proposition 18**

$$A^{-1} = (D - B)^{-1} - \frac{\beta}{1 + \beta\sigma} (D - B)^{-1} e e^T (D - B)^{-1}.$$

*Proof* By Eq. (11). □

**Proposition 19**

$$\epsilon_{k;l} = \left( (D - B)^{-1}e \right)_{k;l} = \frac{1}{(1 - (\alpha + \beta)\tau_k) {}_{k;l}D_{k;l}}.$$

The computation of  $\epsilon$  costs  $O(L)$  flops once  ${}_{k;l}D_{k;l}$  and  $\tau_k$  have been computed.

*Proof* By direct computation with formulas we have already found. □

**Proposition 20** *Let*

$$\psi_{k;l} = \frac{z_{k;l}}{{}_{k;l}D_{k;l}}, \quad \text{and} \quad \mu_k = \sum_{l \in \mathbb{N}_{\lambda_k}} \psi_{k;l}.$$

Then

$$w_{k;l} = \left( (D - B)^{-1}z \right)_{k;l} = \psi_{k;l} + \frac{(\alpha + \beta)\mu_k}{(1 - (\alpha + \beta)\tau_k) {}_{k;l}D_{k;l}}.$$

The cost of computing  $w = (D - B)^{-1}z$  is  $O(L)$  flops once  ${}_{k;l}D_{k;l}$  is available.

*Proof* By direct computation with formulas we have already found. □

## Proposition 21

$$A^{-1}z = w - \frac{\beta(\epsilon^T z)}{1 + \beta\sigma}\epsilon.$$

The cost is  $O(L)$  flops once  $\sigma$ ,  $w$  and  $\epsilon$  have been computed.

*Proof* By direct computation with formulas we have already found.  $\square$

## References

1. Aslam, J.A., Pelekhov, E., Rus, D.: The star clustering algorithm for information organization. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping multidimensional data. Springer, New York (2006)
2. Berkhin, P.: A survey of clustering data mining techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping multidimensional data. Springer, New York (2006)
3. Eppstein, D.: Fast hierarchical clustering and other applications of dynamic closest pairs. *J. Exp. Algorithmics* **5**(1) (2000)
4. Forgy, E.: Cluster analysis of multivariate data: efficiency vs. interpretability of classifications. *Biometrics* **21**(3), 768 (1965)
5. Jain, A.K.: Data clustering: 50 years beyond  $K$ -means. *Pattern Recogn. Lett.* **31**(8), 651–666 (2010)
6. Kohonen, T.: “Self-Organization and Associative Memory,” Springer series in Information Sciences. Springer, New York (1989)
7. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the 5th ACM SIGKDD, pp. 16–22. San Diego (1999)
8. Lindsten, F., Ohlsson, H., Ljung, L.: Just Relax and Come Clustering! A Convexification of  $k$ -Means Clustering. Technical report from Automatic Control at Linköpings universitet, Report no.: LiTH-ISY-R-2992 (2011)
9. Lloyd, S.P.: Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982)
10. Marroquin, J.L., Girosi, F.: Some extensions of the  $k$ -means algorithm for image segmentation and pattern classification. Technical Report A.I. Memo 1390, MIT Press, Cambridge, MA, USA, (1993)
11. McCallum, A., Nigam, K., Ungar, L.H.: Efficient Clustering of High-Dimensional Data Set with Application to Reference Matching. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 169–178 (2000)
12. Ohlsson, H., Ljung, L.: A Convex Approach to Subspace Clustering. In: 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC). Orlando (2011)
13. Ostrovsky, R., Rabani, Y., Schulman, L.J., Swamy, C.: The effectiveness of Lloyd-Type Methods for the  $k$ -Means Problem. In: 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS’06) (2006)
14. Rose, K., Gurewitz, E., Fox, G.C.: A deterministic annealing approach to clustering. *Pattern Recogn. Lett.* **11**, 589–594 (1990)
15. Rousseeuw, P.J., Kaufman, L., Trauwert, E.: Fuzzy clustering using scatter matrices. *Comput. Stat. Data Anal.* **23**, 135–151 (1996)
16. Steinbach, M., Karypis, G., Kumar, V.: A comparison of document clustering techniques. In: Proceedings of the 6th ACM SIGKDD, World Text Mining Conference. Boston (2000)
17. Teboulle, M., Berkhin, P., Dhillon, I., Guan, Y., Kogan, J.: Clustering with Entropy-Like  $k$ -Means Algorithms. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data. Springer, New York (2006)
18. Vattani, A.:  $K$ -means requires exponentially many iterations even in the plane. DCG (2011)
19. Volkovich, Z., Kogan, J., Nicholas, C.: Sampling methods for building initial partitions. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping multidimensional data. Springer, New York (2006)
20. Xu, R., Wunsch, D.: Survey of Clustering Algorithms. *IEEE Trans. Neural Netw.* **16**(3) (2005)
21. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a  $k$ -means clustering algorithm. *Appl. Stat.* (1979)
22. Drake, J., Hamerly, G.: Accelerated  $k$ -means with adaptive distance bounds. In: 5th NIPS workshop on optimization for machine learning (2012)

23. Kanungo, T. et al.: An efficient k-means clustering algorithm: analysis and implementation. *Pattern analysis and machine intelligence, IEEE Transactions* (2002)
24. Elkan, C.: Using the triangle inequality to accelerate k-means. *ICML* **3** (2003)
25. Pelleg, D., Moore, A.W.: X-means: extending K-means with efficient estimation of the number of clusters. *ICML* (2000)
26. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics (2007)
27. Davies, D.L., Bouldin, D.W.: A cluster separation measure. *Pattern Anal. Mach. Intell. IEEE Trans.* **2** (1979)
28. Chen, T.S., et al.: A combined K-means and hierarchical clustering method for improving the clustering efficiency of microarray. *Intelligent Signal Processing and Communication Systems. ISPACS*. In: *Proceedings of 2005 International Symposium on*. IEEE (2005)
29. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP* **2**(1) (2009)

Calcolo is a copyright of Springer, 2017. All Rights Reserved.