A balanced k-means algorithm for weighted point sets

Steffen Borgwardt, Andreas Brieden and Peter Gritzmann

Abstract The classical k-means algorithm for partitioning n points in \( \mathbb{R}^d \) into k subsets is one of the most popular and widely spread clustering methods in scientific and business applications. The present paper gives a generalization that is capable of handling weighted point sets and prescribed lower and upper bounds on the cluster sizes. The new algorithm replaces the assignment step of k-means by the computation of a weight-balanced least-squares assignment. This is modelled as a linear program over a weight-balanced partition polytope whose optimal vertices correspond to clusterings that allow strongly feasible power diagrams. We use this correspondence to derive a worst-case upper bound \( n^{O(dk)} \) for the number of operations. This is similar to the known upper bound for k-means, polynomial for fixed \( k \) and \( d \), and in view of the known complexity results for k-means, essentially the best one can expect. Further, we show the kernelizability of our approach.

Keywords k-means · clustering · least-squares assignments · power diagrams · mathematical programming

Mathematics Subject Classification (2010) 91C20 · 90C90 · 68Q32 · 90C46 · 90C27 · 52B12 · 90C57

Steffen Borgwardt
Fakultät für Mathematik, Technische Universität München, D-80290 München, Germany,
E-mail: borgwardt@ma.tum.de

Andreas Brieden
Fakultät für Wirtschafts- und Organisationswissenschaften, Universität der Bundeswehr München,
D-85579 Neubiberg, Germany, E-mail: andreas.brieden@unibw.de

Peter Gritzmann
Fakultät für Mathematik, Technische Universität München, D-80290 München, Germany,
E-mail: gritzmann@tum.de
1 Introduction

Unrestricted multiclass clustering is a central concept in data analysis. One of the most influential and widely-used clustering method is the $k$-means algorithm (or Lloyd’s algorithm) [20,19]. Given a set of $n$ points and $k$ different initial sites in $\mathbb{R}^d$, it iteratively performs two steps (ignoring some technical details): First, every point is assigned to a closest site. This partitions the points into $k$ clusters, one for each site. Second, the sites are updated to be the arithmetic means of the clusters. These two-step iterations are performed until the sites do not change anymore. The algorithm exhibits many favorable properties, and is well-accepted for its simplicity and fast convergence in practice. Also, it is still subject to quite extensive theoretical analysis motivated by the discrepancy between its excellent behaviour in practice and its known worst-case behaviour; see [4,27].

In the present paper, we present a generalization by introducing weight-balanced $k$-means. Within this framework, we are able to deal with weighted point sets and prescribed upper and lower bounds for the cluster sizes. Weighted point sets and the need to create clusters of prescribed cluster sizes arise in many applications. A well-studied such problem occurs in the consolidation of farmland [12,10]: In an agricultural region, $k$ farmers cultivate $n$ lots. The goal is to improve the cost-effective structure in this region by a combinatorial redistribution of these lots, a voluntary lend-lease agreement after which each farmer can work on larger adjacent components. The lots differ in value, and no farmer would accept a considerable deviation from his original total value during this redistribution. When we represent the lots by points in the Euclidean plane and use their values as weights, we arrive at a clustering problem with bounds on the cluster value for each farmer.

The desire to create clusters of prescribed sizes also often arises when data segmentation is applied to arrive at more homogenous clusters for further statistical analysis. See [9] for an introduction into constrained clustering, and further real-world applications. Note that, in particular, weighted points allow for a natural representation of identical, repeated points in data sets.

In contrast to partitioning an unweighted point set, as facilitated by $k$-means, a weighted point set imposes the need to perform partial membership clustering, where points can be fractionally assigned to more than one cluster. This is unavoidable in general if, e.g., a data set consists of two points of weight three and has to be clustered into three parts of weight two or, if a set of three points of equal weight has to be partitioned into two clusters of equal size.

We set up the generalized $k$-means framework by modelling the computation of a weight-balanced least-squares assignment as a special linear program. Its feasible region, the weight-balanced partition polytope, encodes specific cell decompositions of $\mathbb{R}^d$. Such power diagrams are generalized Voronoi diagrams (see [7] for a survey), and appear in learning as the classifiers derived by alltogether models for multiclass classification; see e.g., [26,31,11,15]. Informally, a classifier is a rule for determining to which of the existing clusters a new point in $\mathbb{R}^d$ should be assigned. In our case, each cluster is associated with exactly one of the cells, and a new point is assigned to the clusters of (one, some, all) cells it lies in.
We use these intimate connections to guarantee termination of our algorithm within \( n^{O(dk)} \) iterations. In order to place this result into perspective note that by [18] there is a similar worst-case bound for classical \( k \)-means. So, in a sense, we obtain our extension at marginal additional cost. Further, the upper bound shows that the algorithms runs in polynomial time for fixed \( k \) and \( d \). This should be contrasted with the \( \mathbb{NP} \)-hardness of the \( k \)-means problem for \( k = 2 \), [1], or \( d = 2 \), [21,22]. Further, while \( k \)-means has polynomial smoothed complexity, [3,4], it may in general require exponentially many iterations even in the plane, [27]. However, there are approximation algorithms based on norm maximization for optimal clustering with quite favorable worst-case error bounds, [13].

We further show how to kernelize our algorithm. In fact, one of the key steps for applying clustering algorithms in machine learning is the ability to integrate the use of kernel functions [24,16,25]. Also, must-link constraints can be easily accommodated.

The present paper is organized as follows. Section 2 will provide the relevant terminology and concepts. In Section 3, we outline our main results. Section 4 then prepares tools from the literature involving least-squares assignments and power diagrams. In Section 5, we present our generalized \( k \)-means algorithm, and prove its correctness and termination. Further, we give the indicated bound on the number of iterations of our algorithm. Section 6 shows the kernelizability of our method, while Section 7 contains some final remarks.

2 Preliminaries

We begin with some standard basic notation.

2.1 Weight-balanced clusterings

Let throughout the present paper \( k, n, d \in \mathbb{N} \) with \( n \geq k \geq 2 \). Let \( X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d \) be a data set of distinct points with associated weights \( \Omega := (\omega_1, \ldots, \omega_n) \in \mathbb{R}^n \) and \( \omega_i > 0 \) for all \( i \leq n \). Without loss of generality, we assume that \( X \) is full-dimensional, i.e., the dimension of its affine hull is \( d \); in particular then \( d + 1 \leq n \). Otherwise we could work in the affine hull of \( X \).

Further, let \( \kappa^- := (\kappa_1^-, \ldots, \kappa_k^-)^T, \kappa^+ := (\kappa_1^+, \ldots, \kappa_k^+)^T \in \mathbb{R}^k \) with \( 0 < \kappa_i^- \leq \kappa_i^+ \) and

\[
\sum_{i=1}^k \kappa_i^- \leq \sum_{j=1}^n \omega_j \leq \sum_{i=1}^k \kappa_i^+.
\]

A (partial membership) \( k \)-clustering \( C := (C_1, \ldots, C_k) \) of \( X \) consists of \( k \) clusters \( C_i \), and is defined by an assignment vector \( y := (y_{11}, \ldots, y_{1n}, \ldots, y_{k1}, \ldots, y_{kn})^T \in [0,1]^{kn} \) of \( X \) with \( \sum_{j=1}^k y_{ij} = 1 \) for all \( j \leq n \). Informally, \( y_{ij} \) is the fraction of the total weight \( \omega_j \) of point \( x_j \) that belongs to \( C_i \). Formally, we set \( C_i := (y_{1i}, \ldots, y_{ni})^T \). The support of the cluster \( C_i \) is \( \text{supp}(C_i) := \{x_j : y_{ij} > 0\} \).
the support of the clustering $C$ is the tuple $\text{supp}(C) := (\text{supp}(C_1), \ldots, \text{supp}(C_k))$.

We use the notation $|C_i| := \sum_{j=1}^{n} y_{ij} \omega_j$ to refer to the total weight or the size of cluster $C_i$. The tuple $|C| := ([C_1], \ldots, [C_k])^T$ is the shape of $C$. The center of gravity $c_i$ of $C_i$ is given by

$$c_i := \frac{1}{\sum_{j=1}^{n} y_{ij} \omega_j} \sum_{j=1}^{n} y_{ij} \omega_j x_j .$$

In this paper we deal with the task of finding optimal weight-balanced clusterings that satisfy $\kappa^- \leq |C| \leq \kappa^+$ componentwise.

2.2 Weight-balanced least-squares assignments

Least-squares assignments are a common kind of clustering, and appear in many real-world applications like facility location or clustering with low variance. They allow intuitive interpretations such as measuring the cost to supply customers with geographic positions $x_1, \ldots, x_n$ from supply sites $s_1, \ldots, s_k \in \mathbb{R}^d$ with respect to a quadratic-loss transport. Several classical clustering algorithms are based on least-squares assignments, a prime example being the $k$-means algorithm, which computes such an assignment in each iteration.

Let $S := \{s_1, \ldots, s_k\}$ be a set of sites in $\mathbb{R}^d$. A clustering $C := (C_1, \ldots, C_k)$ is a weighted $S$-least-squares assignment of $X$ if and only if

$$\sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot \|x_j - s_i\|^2$$

is minimal for all clusterings of $X$. We call $C$ a weight-balanced $(S, \kappa^-, \kappa^+)$-least-squares assignment if it is weight-balanced and minimal with respect to all clusterings of $X$ of shape $\kappa^- \leq |C| \leq \kappa^+$. We will add ‘strict’ to signify that the minimal assignment is unique.

In the degenerate case with $s_i = s_j$ for some $i \neq j$, we do not distinguish between clusters $C_i$ and $C_j$. Rather, we treat them as a single cluster $C_{ij}$ defined by $C_{ij} := (y_{i1} + y_{j1}, \ldots, y_{in} + y_{jn})^T$. The cluster adheres to the size bounds $\kappa^-_{ij} := \kappa^- + \kappa^-$ and $\kappa^+_{ij} := \kappa^+ + \kappa^+$. The strict minimality of the underlying clustering refers to a comparison with all clusterings where $C_i$ and $C_j$ are treated as the combined cluster $C_{ij}$.

2.3 Feasible power diagrams

Power diagrams are special kinds of cell decompositions, and generalize the well-known Voronoi diagrams. In learning, they arise as classifiers derived by altogether models for multiclass classification. See [5] for a survey of this data structure.
A balanced k-means algorithm for weighted point sets

A power diagram is specified by a set of distinct sites \( S := \{ s_1, \ldots, s_k \} \subset \mathbb{R}^d \) and parameters \( \Sigma := (\sigma_1, \ldots, \sigma_k) \in \mathbb{R} \). Using these parameters, the \( i \)-th power cell \( P_i \) is defined by

\[
P_i := \{ x \in \mathbb{R}^d : \| x - s_i \|^2 - \sigma_i \leq \| x - s_j \|^2 - \sigma_j \text{ for all } i \neq j \}.
\]

Here \( \| \cdot \| \) denotes the Euclidean norm. The power diagram \( P \) then is the tuple \( P := (P_1, \ldots, P_k) \).

For our weighted point sets, we say that a power diagram \( P \) is feasible for the clustering \( C \) if \( \text{supp}(C_i) \subset P_i \) for all \( i \leq k \). Then \( C \) allows \( P \). More strongly, \( P \) supports \( C \) if \( \text{supp}(C_i) = P_i \cap X \) for all \( i \leq k \).

In this paper, a special kind of feasible power diagrams, the so-called strongly feasible power diagram \( P \) is important, [14], that supports \( C \) and has the following additional property. Let \( G(C) \) be the multigraph with vertices \( C_1, \ldots, C_k \) and an edge labeled with \( x_j \) incident to \( C_i \) and \( C_l \) with \( i \neq l \) if and only if \( x_j \in \text{supp}(C_i) \cap \text{supp}(C_l) \). Then \( G(C) \) does not contain a cycle with two or more different edge labels.

If \( P \) is a feasible power diagram for \( C \), and the sites coincide with the centers of gravity of the clusters, i.e. \( s_i = c_i \) for all \( i \leq k \), then \( P \) is called a centroidal power diagram.

Note that a power diagram requires distinct sites by definition. We treat the degenerate case with \( s_i = s_j \) for some \( i \neq j \) again by combining the associated clusters \( C_i \) and \( C_j \). A feasible power diagram for this clustering does not contain separate cells \( P_i \) and \( P_j \), but a single cell \( P_{ij} \) such that \( \text{supp}(C_{ij}) \subset P_{ij} \). In the same sense, we talk about strongly feasible power diagrams and centroidal power diagrams with cells \( P_{ij} \) for \( C_{ij} \).

3 Main results

We present a generalization of the \( k \)-means algorithm that can handle weighted point sets and prescribed lower and upper bounds on the cluster sizes. More precisely, Algorithm 3 accepts weighted points, sites and lower and upper bounds for the cluster sizes and computes a strongly feasible centroidal weight-balanced least-squares assignment.

While the termination of the classical \( k \)-means algorithm is clear (when appropriate precautions are taken if empty cells occur) because none of the finitely many clusterings is visited twice, finiteness of Algorithm 3 is not obvious. In fact, since we deal with partial membership clusterings, there is an infinite number of possible states! However, our first result proves termination.

**Theorem** Algorithm 3 terminates with a clustering that allows a strongly feasible centroidal power diagram.

Inaba et al. [18] proved an upper bound of \( n^{O(kd)} \) for the worst-case running time of the classical \( k \)-means algorithm, by bounding the total number of least-squares assignments of a data set. See also [17,23] for other bounds on the number of related types of clusterings. We derive a similar upper bound in our much more general
framework. More precisely, with $e$ denoting the Euler number we obtain the following bound on the number of iterations of our algorithm which is polynomial for fixed dimension and fixed number of clusters.

**Theorem**  The number of iterations of Algorithm 3 is bounded by $(40ek^2n)^{(d+1)k-1}$.

The proofs of these theorems are given in Section 5.2 (as Theorems 1 and 3). In Section 6, we show the kernelizability of our approach. The kernelized versions of the computation of a weight-balanced least-squares assignment and of our $k$-means framework are stated in Algorithms 4 and 5. The bound of Theorem 3 on the running time transfers to the kernelized approach, with the dimension $d$ being the dimension $d_\phi$ of the inner product space the kernel maps to.

### 4 Weight-balanced least-squares assignments

In this section, we discuss how to compute a weight-balanced least-squares assignment for a given data set $X$ with weights $\Omega$, a set of sites $S$ and cluster size bounds $\kappa^\pm \in \mathbb{R}_k$. At several places of this paper, we make heavy use of the work of [14], where weight-balanced least-squares assignments and their relation to power diagrams are studied. Rather than assuming detailed knowledge of these results, we repeat the relevant details now in our notation and state their implications concisely, to make the paper self-contained.

#### 4.1 A linear programming formulation

We are interested in a weight-balanced $(S, \kappa^-, \kappa^+)$-least-squares assignment of $X$. We can describe the task to find such an assignment as an optimization problem in the form

$$\min \sum_{i=1}^k \sum_{j=1}^n y_{ij} \omega_j \cdot \|x_j - s_i\|^2 \quad \text{s.t.} \quad \kappa^- \leq |C_i| \leq \kappa^+ \quad (i \leq k).$$

We use the variables $y_{ij} \in [0, 1]$ to indicate how much of the weight $\omega_j$ of point $x_j$ is associated to cluster $C_i$. Then the constraints can be described by the set

$$\kappa_i^- \leq \sum_{j=1}^n y_{ij} \omega_j \leq \kappa_i^+ \quad (i \leq k)$$

$$\sum_{i=1}^k y_{ij} = 1 \quad (j \leq n)$$

$$y_{ij} \geq 0 \quad (i \leq k, j \leq n)$$

of linear equalities. The first line in this system implies that the clusters satisfy the given bounds, whereas the second line guarantees that each point is fully assigned. Note that these constraints define a polytope; we call it the **weight-balanced partition polytope**.
A balanced k-means algorithm for weighted point sets

The objective function can be written as

$$\min_{k} \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot \|x_j - s_i\|^2 = \min_{k} \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot (x_j^T x_j - 2 x_j^T s_i + s_i^T s_i)$$,

which is linear in the $y_{ij}$, as $x_j$, $s_i$ and $\omega_j$ are fixed. Note further that

$$\sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot (x_j^T x_j - 2 x_j^T s_i + s_i^T s_i) = \sum_{j=1}^{n} \omega_j \cdot (x_j^T x_j - 2 x_j^T s_i + s_i^T s_i)$$.

Thus, the objective function can be simplified to

$$\min_{k} \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot (s_i^T s_i - 2 x_j^T s_i)$$.

Algorithm 1 and the subsequent lemma sum up this construction.

**Algorithm 1** Weight-balanced least-squares assignment

- **Input:** $d, k, n \in \mathbb{N}, X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d, \Omega := (\omega_1, \ldots, \omega_n), S := \{s_1, \ldots, s_k\} \subset \mathbb{R}^d, \kappa^- \leq \kappa^+ \in \mathbb{R}^d$
- **Output:** an $(S, \kappa^-, \kappa^+)$-least-squares assignment of $X$

Solve the linear program

$$\min_{\kappa^-} \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot (s_i^T s_i - 2 x_j^T s_i)$$

$$\kappa^- \leq \sum_{j=1}^{n} y_{ij} \omega_j \leq \kappa^+ \quad (i \leq k)$$

$$\sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} = 1 \quad (j \leq n)$$

$$y_{ij} \geq 0 \quad (i \leq k, j \leq n)$$.

and return a basic feasible solution as the assignment.

**Lemma 1** Algorithm 1 computes a weight-balanced least-squares assignment by linear programming with $k \cdot n$ variables and $(k + 1) \cdot n + 2 \cdot k$ constraints.

4.2 Strongly feasible power diagrams

When varying the sites, the different outputs of Algorithm 1 constitute a special subclass $\mathcal{V}$ of the vertices of the weight-balanced partition polytope. For our purposes we need a characterization of $\mathcal{V}$ in terms of strongly feasible power diagrams.

Feasible power diagrams are intimately related to least-squares assignments of unweighted point sets. [6] proved that if $C$ is a balanced least-squares assignment of an unweighted data set (i.e. $\Omega = (1, \ldots, 1)$) and $\kappa^- = \kappa^+$, then $C$ allows a strongly feasible power diagram. It is also possible to obtain this statement by a careful interpretation of Theorem 5 in [8]. These results allow a far-reaching extension, [14], leading, in particular, to a complete characterization in the case $\kappa^- = \kappa^+$. Here we only need the following particular result (see [14], Corollaries 2.2, 2.3 (including their proofs), and Lemmata 4.2, 4.3).
Proposition 1 ([14]) Let $X$ be a weighted data set, and let $C$ be a (strict) weight-balanced least-squares assignment for $X$. Then $C$ allows a (strongly) feasible power diagram.

Further, if $C$ allows a strongly feasible power diagram, its assignment vector $y$ contains at most $2(k - 1)$ fractional components, i.e., components with $0 < y_{ij} < 1$.

In particular, if $C$ corresponds to a vertex in $\mathcal{V}$ then, by Proposition 1, $C$ allows a strongly feasible power diagram.

Corollary 1 Algorithm 1 computes a clustering $C$ that allows a strongly feasible power diagram.

Proof By Lemma 1, Algorithm 1 computes a weight-balanced least-squares assignment $y$ and returns a vertex of the weight-balanced partition polytope. By Proposition 1, it corresponds to a clustering $C$ that allows a strongly feasible power diagram. $\square$

5 Weight-balanced $k$-means

The classical $k$-means algorithm is one of the most widely used clustering algorithms. As a service to the reader, we state a basic version as Algorithm 2.

Algorithm 2 $k$-means

| Input: $d, k, n \in \mathbb{N}, X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d, S := \{s_1, \ldots, s_k\} \subset \mathbb{R}^d$ |
| Output: a clustering $C = (C_1, \ldots, C_k)$ of $X$ and the arithmetic means $c_1, \ldots, c_k$ as sites |

1. Partition $X$ into a clustering $C = (C_1, \ldots, C_k)$ by assigning $x_j \in X$ to a cluster $C_i$ that is closest to site $s_i \in S$.
2. Update each site $s_i$ as the center of gravity of cluster $C_i$; if $|C_i| = 0$, choose $s_i = x_l$ for a random $l \leq n$ with $x_l \neq s_j$ for all $j \leq k$. If the sites change, go to (1.).

5.1 Balanced $k$-means for weighted point sets

By replacing the trivial assignment step (1.) with the computation of a strongly feasible weight-balanced clustering according to Algorithm 1, we can generalize this framework to deal with weighted point sets and lower and upper bounds on the cluster sizes. Algorithm 3 describes this in pseudocode.

5.2 Correctness and termination

It is easy to show correctness and termination of the standard $k$-means algorithm. The trivial assignment of each point to a closest site in each iteration is readily interpreted as the computation of an unconstrained least-squares assignment. A straightforward
Algorithm 3 weight-balanced k-means

- Input: \( d, k, n \in \mathbb{N}, X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d, \Omega := (\omega_1, \ldots, \omega_k), S := \{s_1, \ldots, s_k\} \subset \mathbb{R}^d, \kappa^* \leq \kappa^* \in \mathbb{R}^d \)
- Output: strongly feasible centroidal weight-balanced least-squares assignment of \( X \) and the centers of gravity as sites

1. Apply Algorithm 1 for the current set of sites to obtain assignment \( y \).
2. Update each site \( s_i \) as the center of gravity \( c_i := \frac{1}{\sum_j \omega_j} \sum_j \omega_j s_j \).

If the objective function value decreased during the last iteration, go to (1); else return the current assignment and sites.

---

Theorem 1 Algorithm 3 terminates with a clustering that allows a strongly feasible centroidal power diagram.

Proof First, we prove that the center of gravity \( c_i \) of a cluster is an optimal site with respect to a fixed weight-balanced least-squares assignment. Hence, if it terminates, the algorithm produces a clustering with a feasible centroidal power diagram.

So, let \( C := \{C_1, \ldots, C_k\} \) be a fixed clustering with centers of gravity \( c_1, \ldots, c_k \), and let \( s_1, \ldots, s_k \) be optimal sites. We set \( s_i = c_i - z_i \) for \( i \leq k \). For each \( C_i \), we then have

\[
\sum_{j=1}^{n} y_{ij} \omega_j \|x_j - s_i\|^2 =
\]

\[
= \sum_{j=1}^{n} y_{ij} \omega_j x_j^T x_j - 2 \sum_{j=1}^{n} y_{ij} \omega_j (c_i - z_i)^T (c_i - z_i) + \sum_{j=1}^{n} y_{ij} \omega_j (c_i - z_i)^T (c_i - z_i)
\]

\[
= \sum_{j=1}^{n} y_{ij} \omega_j x_j^T x_j - 2 |C_i| c_i^T c_i + 2 |C_i| c_i^T z_i + |C_i| c_i^T c_i - 2 |C_i| c_i^T z_i + |C_i| z_i^T z_i
\]

\[
= \sum_{j=1}^{n} y_{ij} \omega_j x_j^T x_j - |C_i| c_i^T c_i + |C_i| z_i^T z_i.
\]

Since the clustering is fixed, and since \( c_i^T z_i \geq 0 \), this sum is minimal for \( z_i = 0 \), respectively for \( s_i = c_i \).
Let
\[ \Theta(C, S) := \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \omega_j \cdot (s_i^T s_i - 2x_j^T s_i) \]
be the least-squares value for the clustering \( C \) and sites \( S \), let \( C^{(l)} \) be the returned optimal clustering for the \( l \)-th iteration, and let \( S^{(l)} \) be the set of sites used for the linear program to produce \( C^{(l)} \). Note that \( S^{(l+1)} \) consists of the centers of gravity of the clusters of \( C^{(l)} \).

The above computation shows that the sequence \( \Theta(C^{(l)}, S^{(l)}) \) is monotonously decreasing due to
\[ \Theta(C^{(l)}, S^{(l)}) \geq \Theta(C^{(l)}, S^{(l+1)}) \geq \Theta(C^{(l+1)}, S^{(l+1)}) \]
The termination criterion in (2.) is thus well-defined, and implies that the sequence is strictly decreasing until termination. By the final update of the sites as centers of gravity in (2.), we return a feasible centroidal power diagram.

In each iteration we compute a clustering corresponding to a vertex in \( \mathcal{V} \). By Corollary 1, it allows a strongly feasible power diagram. Since there are only finitely many vertices and the objective function is strictly decreasing, no vertex is visited twice. This proves termination. \( \square \)

In the next section, we provide a worst-case upper bound on the number of iterations of our algorithm.

5.3 A bound on the number of iterations for strongly weight-balanced clustering

The first step towards a bound on the number of iterations of Algorithm 3 is to bound the number of different supports of strongly feasible power diagrams for weight-balanced clusterings. This will be done by estimating the number of different point-cell incidence structures that can possibly be realized by power diagrams. To be more precise, for a power diagram \( P := (P_1, \ldots, P_k) \) let
\[ \mathcal{X}(P) := (X \cap P_1, \ldots, X \cap P_k) \]
be the \((X, P)\)-incidence pattern. In general i.e., if we do not want to particularly stress a specific power diagram, we will speak of a power pattern. Note that for a weight-balanced least-squares assignment \( C \) and a corresponding strongly feasible power diagram \( P := (P_1, \ldots, P_k) \) we have
\[ \text{supp}(C) = (\text{supp}(C_1), \ldots, \text{supp}(C_k)) = (X \cap P_1, \ldots, X \cap P_k) = \mathcal{X}(P) \]
Of course, since our definition of an \((X, P)\)-incidence pattern does not involve weight-balancing, the number of power patterns will in general be larger than the number of strongly feasible power diagrams for weight-balanced clusterings.

In order to provide an upper bound for the number of different power patterns, we use a well-known bound on the number of so-called sign-patterns of a set of polynomials by Warren [30]. (In [30] only a bound for \([-1, 1]\)-sign patterns is given;
this can, however, easily be extended to the \(-1, 0, 1\)-sign patterns that we need here; see e.g. \([2]\).

Let \(p_1, \ldots, p_t\) be a system of real polynomials in \(s\) variables. For a point \(z \in \mathbb{R}^s\), the sign-pattern of \(p_1, \ldots, p_t\) is a tuple \(v(z) = (v_1, \ldots, v_t)^T \in \{+1, 0, -1\}^t\) defined by \(v_i = -1\) if \(p_i(z) < 0\), \(v_i = 0\) if \(p_i(z) = 0\) and \(v_i = 1\) if \(p_i(z) > 0\).

**Proposition 2** (\([30]\)) Let \(p_1, \ldots, p_t\) be a system of real polynomials in \(s\) variables, all of degree at most \(l\). If \(s \leq 2t\), then the number of different sign-patterns of this system is bounded above by

\[
\left( \frac{8e \cdot l \cdot t}{s} \right)^s
\]

We will now use Proposition 2 to give an upper bound on the number of different power patterns.

**Theorem 2** There are at most

\[
\left( \frac{8e \cdot (k - 1)n}{d} \right)^{(d+1)k-1}
\]

different power patterns.

**Proof** An \((S, \Sigma)\)-power diagram with \(k\) cells in \(\mathbb{R}^d\) is defined by \(k\) distinct sites \(S := \{s_1, \ldots, s_k\} \subset \mathbb{R}^d\) and \(k\) weights \(\Sigma := \{\sigma_1, \ldots, \sigma_k\} \subset \mathbb{R}\). Its cells are the same as those of an \((S, \Sigma + \sigma')\)-power diagram, where \(\Sigma + \sigma' = \{\sigma_1 + \sigma', \ldots, \sigma_k + \sigma'\}\) for some \(\sigma' \in \mathbb{R}\). Due to this, one may choose \(\sigma_k = 0\), and it suffices to use a vector

\[
g = (s_1, \sigma_1, \ldots, s_{k-1}, \sigma_{k-1}, s_k)^T \in \mathbb{R}^{(d+1)k-1}
\]
to define the cells of the \((S, \Sigma)\)-power diagram. We call these vectors reduced control vectors.

For any pair \(i \neq j\) and any point \(x \in X\), we define an algebraic surface of reduced control vectors in \(\mathbb{R}^{(d+1)k-1}\) via

\[
\|x - s_i\|^2 - \sigma_i = \|x - s_j\|^2 - \sigma_j
\]

(Recall that we use \(\sigma_k = 0\).) These \(\binom{k}{2}\) surfaces are of degree 2.

Two reduced control vectors \(g, g'\) will be regarded equivalent if the incidence patterns of the corresponding power diagrams \(P, P'\) coincide i.e., \(\mathcal{X}(P) = \mathcal{X}(P')\). Note that this defines an equivalence relation on the vectors in \(\mathbb{R}^{(d+1)k-1}\).

The above surfaces yield an arrangement in \(\mathbb{R}^{(d+1)k-1}\), i.e., a decomposition of \(\mathbb{R}^{(d+1)k-1}\) into relatively open connected cells of dimension \(0, 1, \ldots, (d + 1)k - 1\), where each cell is the inclusion-maximal connected set of points in the intersection of some subset of these surfaces. Note that the condition \(\|x - s_i\|^2 - \sigma_i = \|x - s_j\|^2 - \sigma_j\) implies that \(x\) is on the common boundary of cells \(P_i\) and \(P_j\) in the associated power diagram. Thus the arrangement induced by the above surfaces is a representation of precisely these equivalence classes of reduced control vectors providing the same incidence pattern.
Our \( \binom{n}{2} \) surfaces yield a set of polynomials \( p_1, \ldots, p_{\binom{n}{2}} \). The sign patterns of the reduced control vectors \( z \in \mathbb{R}^{(d+1)k-1} \) are in one-to-one correspondence with the faces of our arrangement.

We now apply Proposition 2. We have \( \binom{n}{2} \) polynomials, all of them are of degree 2, and the arrangement lies in \( \mathbb{R}^{(d+1)k-1} \). In our case, the condition \( s \leq 2t \) translates to \( (d+1)k-1 \leq \binom{d}{2} n \). It is satisfied since from our general assumptions \( d+1 \leq n \) and \( k \geq 2 \) we obtain
\[
(d+1)k-1 < (d+1)k \leq nk \leq (k-1)kn.
\]

Hence, we get the upper bound
\[
\left( \frac{8e \cdot 2 \binom{n}{2} n}{(d+1)k-1} \right)^{(d+1)k-1} \leq \left( \frac{8e \cdot k(k-1)n}{(d+1)k-k} \right)^{(d+1)k-1} = \left( \frac{8e \cdot (k-1)n}{d} \right)^{(d+1)k-1}
\]
which proves the assertion. \( \square \)

**Theorem 3** The number of iterations of Algorithm 3 is bounded by
\[
(40ek^2n)^{(d+1)k-1}.
\]

**Proof** By Corollary 1, in each iteration, the linear program of the algorithm computes a clustering that allows a strongly feasible power diagram.

The sequence of objective function values of these linear programs is strictly decreasing. Thus, the number of iterations is bounded above by the number \( |Y| \) of vertices of the weight-balanced partition polytope corresponding to clusterings that allow a strongly feasible power diagram.

Given a clustering \( C \) that allows a strongly feasible power diagram, we will bound the number of vertices of the weight-balanced partition polytope whose clusterings share the same pattern. Combining this number with the bound in Theorem 2 then yields the claim.

Let \( y^* \) be the assignment vector of such a clustering \( C \). By Proposition 1, \( y^* \) contains at most \( 2(k-1) \) entries \( 0 < y_{ij}^* < 1 \). This implies that at most \( k-1 \) points belong to more than one cluster.

Let \( X_{\text{int}}(y^*) \) denote the subset of \( X \) of all points \( x_j \) that belong to exactly one cluster. In the following linear system of inequalities all points of \( X_{\text{int}}(y^*) \) are fixed.

\[
\begin{align*}
\kappa_i^- \leq & \sum_{j=1}^n \omega_j \cdot y_{ij} \leq \kappa_i^+ & (i \leq k) \\
\sum_{i=1}^k y_{ij} = & 1 & (j \leq n) \\
y_{ij} = & y_{ij}^* & (i \leq k, j \leq n : y_{ij}^* \in \{0,1\}) \\
y_{ij} \geq & 0 & (i \leq k, j \leq n : y_{ij}^* \notin \{0,1\}).
\end{align*}
\]

This system encodes the **restricted weight-balanced partition polytope** \( Q(y^*) \) as the intersection of the weight-balanced partition polytope with the set of hyperplanes \( y_{ij} = y_{ij}^* \) for all \( y_{ij}^* \in \{0,1\} \). It represents all feasible clusterings \( C' := \{C_1', \ldots, C_k'\} \) in which all points of \( X_{\text{int}}(y^*) \) are integrally assigned i.e.,
\[
x_j \in X_{\text{int}}(y^*) \land \exists i,l : x_j \in \text{supp}(C_i') \cap \text{supp}(C_l') \Rightarrow i = l.
\]
Note that \( y^* \), being a vertex of the weight-balanced partition polytope, is also a vertex of the polytope \( Q(y^*) \).

Using the parameter vector \( z := (z_{11}, \ldots, z_{kn})^T \in \mathbb{R}^{kn} \) defined by \( z_{ij} := 1 \) if \( y_{ij}^* = 1 \), and \( z_{ij} := 0 \) otherwise, \( Q(y^*) \) is equivalent to the polytope \( Q'(y^*) \)

\[
\kappa^*_i - \sum_{j \neq y_i^*} \omega_j \cdot z_{ij} \leq \sum_{y_i^* \neq (0,1)} \omega_j y_{ij} \leq \kappa^*_i - \sum_{j \neq y_i^*} \omega_j \cdot z_{ij} \quad (i \leq k)
\]

\[
\sum_{y_i^* \neq (0,1)} y_{ij} = 1 - \sum_{i=1}^k z_{ij} \quad (j \leq n : \exists y_{ij}^* \notin \{0,1\})
\]

\[
y_{ij} \geq 0 \quad (i \leq k, j \leq n : y_{ij}^* \notin \{0,1\}).
\]

As there are at most \( 2(k-1) \) values \( 0 < y_{ij}^* < 1 \), there are at most \( 2(k-1) \) variables, and thus \( Q'(y^*) \) has dimension at most \( \mathbb{R}^{2(k-1)} \). Further, \( Q'(y^*) \) is presented by at most \( 5k-2 \) constraints: line 1 contributes at most \( 2k \) constraints, line 2 at most \( k-1 \), and line 3 at most \( 2(k-1) \).

The number of vertices of the weight-balanced partition polytope that correspond to such clusterings \( C' \) and that allow a strongly feasible power diagram is bounded above by the number of vertices of \( Q'(y^*) \). This number does not exceed the number of bases for the system of inequalities defining \( Q'(y^*) \). The latter is at most \( \binom{5k^2-2}{2(k-1)} \leq (5k)^{2k-2} \).

By Theorem 2, the number of power patterns is at most \( \frac{8e \cdot (k-1) n}{d} (d+1)^{k-1} \). As pointed out before, this is also an upper bound for the number of supports of strongly feasible power diagrams. For each of these supports, the restricted weight-balanced partition polytope has at most \( (5k)^{2k-2} \) vertices. Since, during a run of Algorithm 3, no vertex is visited twice, we obtain the upper bound

\[
(5k)^{2k-2} \cdot \left( \frac{8e \cdot (k-1) n}{d} \right)^{(d+1)^{k-1}} \leq \left( \frac{5k \cdot 8e \cdot (k-1) n}{d} \right)^{(d+1)^{k-1}} \leq (40e^2 n)^{(d+1)^{k-1}}
\]

on the number of its iterations. \( \square \)

As the proof of Theorem 3 shows, the number of different strongly feasible power diagrams is bounded by the same number. Hence, with the aid of Proposition 1 we obtain the following corollary.

**Corollary 2** For given \( X, \Omega \) and \( \kappa^k \), the number of strict weight-balanced least-squares assignments is bounded by

\[
(40e^2 n)^{(d+1)^{k-1}}.
\]

### 6 Kernelizability

In many applications of data analysis, e.g., in computational biology, it is important to be able to integrate the use of a **kernel function**. Geometrically, one changes the representation of a data set by mapping it to a higher-dimensional inner product space.
But instead of working with this different representation of the data set, a kernelizable algorithm replaces all inner products of points of the original input by values of a kernel function.

A kernelizable algorithm allows the comparison of results for different representations (by the use of different kernel functions) of a data set, without any conceptual change to the actual algorithm. The well-known kernel \( k \)-means algorithm is a common extension of classical k-means. For some basics about kernel functions, and the arguments we use here; see e.g. \cite{28}. We now show that our generalized \( k \)-means framework is kernelizable, as well.

Let \( \varphi : \mathbb{R}^d \rightarrow V \) map the data space \( \mathbb{R}^d \) to a higher-dimensional inner product space \( V \). As is common practice, we can interpret a kernelized version of our algorithms as a direct application in the inner-product space \( V \), without any conceptual change – we just have to replace all vectors \( s_i \) and \( x_j \) by \( \varphi(s_i) \) and \( \varphi(x_j) \). We define the kernel function \( \Phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \) by \( \Phi(u, v) = \varphi(u)^T \varphi(v) \) for all \( u, v \in \mathbb{R}^d \).

For Algorithm 1, it is particularly simple to produce a kernelized version: The weight-balanced \( k \)-means algorithm is a com- change to the actual algorithm. The well-known kernel \( k \)-means algorithm is a common extension of classical \( k \)-means. For some basics about kernel functions, and the arguments we use here; see e.g. \cite{28}. We now show that our generalized \( k \)-means framework is kernelizable, as well.

Let \( \varphi : \mathbb{R}^d \rightarrow V \) map the data space \( \mathbb{R}^d \) to a higher-dimensional inner product space \( V \). As is common practice, we can interpret a kernelized version of our algorithms as a direct application in the inner-product space \( V \), without any conceptual change – we just have to replace all vectors \( s_i \) and \( x_j \) by \( \varphi(s_i) \) and \( \varphi(x_j) \). We define the kernel function \( \Phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \) by \( \Phi(u, v) = \varphi(u)^T \varphi(v) \) for all \( u, v \in \mathbb{R}^d \).

For Algorithm 1, it is particularly simple to produce a kernelized version: The points in \( X \) and the sites \( S \) only appear in the objective function in the inner products of type \( x_i^T s_j \) or \( s_i^T s_i \). By preprocessing the \( k \cdot n + k \cdot k = k(n + k) \) values \( \Phi(x_j, s_i) \) and \( \Phi(s_i, s_i) \), we obtain the kernelized Algorithm 4.

**Algorithm 4** Weight-balanced kernel least-squares assignment

- **Input:** \( d, k, n \in \mathbb{N}, X := \{ x_1, \ldots, x_n \} \subseteq \mathbb{R}^d, \Omega := \{ \omega_1, \ldots, \omega_k \}, S := \{ s_1, \ldots, s_k \} \subseteq \mathbb{R}^d, k^- \leq k^+ \in \mathbb{R}^k, \) kernel function \( \Phi : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R} \)
- **Output:** an \((S, k^-, k^+)\)-least-squares assignment of \( X \), based on \( \Phi \)

Compute the values \( \Phi(x_j, s_i) \) and \( \Phi(s_i, s_i) \) for all \( i \leq k, j \leq n \).

Solve the linear program

\[
\begin{align*}
\min & \sum_{j=1}^k \sum_{i=1}^n \omega_j \cdot y_{ij} \cdot (\Phi(s_i, s_i) - 2\Phi(x_j, s_i)) \\
\quad & \kappa^- \leq \sum_{j=1}^k \omega_j \cdot y_{ij} \leq \kappa^+ \quad (i \leq k) \\
\quad & \sum_{j=1}^n y_{ij} = 1 \quad (j \leq n) \\
\quad & y_{ij} \geq 0 \quad (i \leq k, j \leq n).
\end{align*}
\]

and return a basic feasible solution as the assignment.

Analogously, Algorithm 3 can be kernelized to become Algorithm 5. For the computation of the centers of gravity we have to use the function \( \varphi \), as they refer to points in space \( V \); they are denoted by \( c_i^\varphi \). (Note that the corresponding centers \( c_i \) in \( \mathbb{R}^d \) need not to be computed; in each iteration, we just need the values \( \Phi(x_j, s_i) = \varphi(x_j)^T \varphi(s_i) \) and \( \Phi(s_i, s_i) = \varphi(s_i)^T \varphi(s_i) \).)

Let \( d_\varphi \) denote the dimension of \( V \) for a given \( \varphi \). By Theorem 3, we can bound the number of iterations of Algorithm 5 as follows.

**Corollary 3** The number of iterations of Algorithm 3 is at most \((40e k^2 n)^{(d_\varphi + 1)k - 1}\).
Algorithm 5 Weight-balanced kernel k-means

- **Input**: \(d, k, n \in \mathbb{N}, X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d, \Omega := \{\omega_1, \ldots, \omega_n\}, S := \{s_1, \ldots, s_k\} \subset \mathbb{R}^d, \kappa^- \leq \kappa^+ \in \mathbb{R}, \phi : \mathbb{R}^d \to V \) and kernel function \(\Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}\) with \(\Phi(u, v) = \phi(u)^T \phi(v)\) for all \(u, v \in \mathbb{R}^d\).
- **Output**: strongly feasible centroidal weight-balanced least-squares assignment of \(X\) in \(V\), based on \(\phi\), and the centers of gravity \(\phi(c_i)\) as sites

1. Apply Algorithm 4 for \(\phi(s_1), \ldots, \phi(s_k)\) to obtain assignment \(y\).
2. Update each site \(\phi(s_i)\) as the center of gravity \(c_i^{\phi} = \frac{1}{\sum_{j=1}^n y_{ij} \omega_j} \sum_{j=1}^n y_{ij} \omega_j \phi(x_j)\). If the objective function value decreased during the last iteration, go to (1.), else return the current assignment and sites.

7 Final Remarks

We close the paper with some remarks.

First, note that in each iteration of Algorithm 3 the feasibility region of the linear program in Step (1.) is the same. An optimal assignment \(y^*\) stays therefore feasible for the subsequent iteration and can hence be used for a warm start. If the centers of gravity did not change significantly, the new optimal assignment \(y^{**}\) will be only a few primal simplex pivot steps away from \(y^*\). Of course, the same is true for Algorithm 5.

Second, in many applications, it is important to respect **must-link constraints** on points of the given data set. These force pairs of points have to be assigned to the same cluster, and form groups of "connected must-link points" by the transitivity of this property. In our context a most natural way of defining must-link constraint for a pair of points \((x_i, x_j)\) in an assignment vector \(y\) is to require that \(y_{il} = y_{jl}\) for all \(l \leq k\).

There are different ways of extending the classical \(k\)-means algorithm to respect such constraints for partitions of unweighted points; see, e.g., [29]. In our model must-link constraints do not require any conceptual change. In fact, let \(I\) denote the set of indices of those points of \(X\) that have to be assigned to the same cluster. In a preprocessing step, we then replace these points by a single convex combination \(x_I\) with weight \(\omega_I\) according to

\[
\omega_I := \sum_{i \in I} \omega_i, \quad x_I := \frac{1}{\omega_I} \sum_{i \in I} \omega_i x_i.
\]

Then we perform our algorithms for the new data. When a vector \(y\) is returned, we can recover a solution in the original setting by defining \(y_{il} := y_{iI}\) for all \(i \in I\) and \(l \leq k\).

Third, in general \(k\)-means and weight-balanced \(k\)-means share the property to compute only a local optimum for the problem of finding best sites. While the computation of a global optimum is \(\text{NP}\)-hard, [1], [22], there are, however, polynomial-time approximation algorithms even in the weight-balanced case with worst-case error bound that depends only on \(d\) and \(k\). In fact, [13] gives a much more general analysis within the framework of norm-maximization. As far as we know, the complexity of computing any local optimum of the \(k\)-means problem is still open.
References

1. D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. Machine Learning, 75:245–249, 2009.
2. N. Alon. Tools from higher algebra. In R.L. Graham, M. Grötschel, and L. Lovász, editors, Handbook of Combinatorics (vol. 2), pages 1749–1783. MIT Press, 1995.
3. D. Arthur, B. Manthey, and H. Röglin. k-Means has polynomial smoothed complexity. In Proc. 50th Ann. IEEE Symposium on Foundations of Computer Science, FOCS ’09, pages 405–414. IEEE Computer Society, 2009.
4. D. Arthur, B. Manthey, and H. Röglin. Smoothed analysis of the k-means method. J. Assoc. Comput. Mach., 58(5):19:1–19:31, 2010.
5. F. Aurenhammer. Power diagrams: Properties, algorithms and applications. SIAM J. Computing, 16(1):78–96, 1987.
6. F. Aurenhammer and R. Klein. Voronoi diagrams. In J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, pages 201–290. Elsevier Science, 1999.
7. A. Brieden and P. Gritzmann. A quadratic optimization model for the consolidation of farmland by means of lend-lease agreements. In D. Ahn, R. Fahrion, M. Oswald, and G. Reinelt (eds.) Operations Research Proceedings 2003: Selected Papers of the International Conference on Operations Research (OR 2003), pages 324–331, 2004.
8. A. Brieden and P. Gritzmann. On clustering bodies: Geometry and polyhedral approximation. Discrete Computational Geometry, 44(3):508–534, 2010.
9. A. Brieden and P. Gritzmann. On optimal weighted balanced clusterings: Gravity bodies and power diagrams. SIAM J. Discrete Mathematics, 26:415–434, 2012.
10. K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. J. Machine Learning Research, 2:265–292, 2001.
11. M. Girolami. Mercer kernel-based clustering in feature space. IEEE Transactions on Neural Networks, 13(3):780–784, 2002.
12. M. Inaba, N. Katoh, and H. Imai. Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering. In Proceedings of the tenth Annual Symposium on Computational Geometry, SCG ’94, pages 332–339, 1994.
13. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. Machine Learning, 75:245–249, 2009.
14. M. Mahajan, P. Nimbhorkar, and K. Varadarajan. The planar k-means problem is NP-hard. In S. Das and R. Uehara, editors, WALCOM: Algorithms and Computation (Proc. 3rd Intern. Workshop Algorithms and Computation), pages 274 – 285. Springer-Verlag, 2009.
15. S. Onn and L. J. Schulman. The vector partition problem for convex objective functions. Mathematics of Operations Research, 26(3):583–590, 2001.
16. B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10(5):1299–1319, 1998.
25. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

26. V. Vapnik. *Statistical learning theory*. Wiley, 1998.

27. A. Vattani. K-Means requires exponentially many iterations even in the plane. *Discrete Computational Geometry*, 45:596–616, 2011.

28. J.-P. Vert, K. Tsuda, and B. Schölkopf. A primer on kernel methods. In *Kernel Methods in Computational Biology*, pages 35–70. MIT Press, 2004.

29. K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the 17th International Conference on Machine Learning*, pages 577–584, 2001.

30. H. E. Warren. Lower bounds for approximation by nonlinear manifolds. *Transactions of the American Mathematical Society*, 133(1):167–178, 1968.

31. J. Weston and C. Watkins. Multi-class support vector machines. Technical report, University of London, 1998.