Turning Big data into tiny data: Constant-size coresets for k-means, PCA and projective clustering

Dan Feldman* Melanie Schmidt† Christian Sohler†

Abstract

We prove that the sum of the squared Euclidean distances from the n rows of an $n \times d$ matrix A to any compact set that is spanned by k vectors in $\mathbb{R}^d$ can be approximated up to $(1+\varepsilon)$-factor, for an arbitrary small $\varepsilon > 0$, using the $O(k/\varepsilon^2)$-rank approximation of A and a constant. This implies, for example, that the optimal k-means clustering of the rows of A is $(1+\varepsilon)$-approximated by an optimal k-means clustering of their projection on the $O(k/\varepsilon^2)$ first right singular vectors (principle components) of A.

A (j,k)-coreset for projective clustering is a small set of points that yields a $(1+\varepsilon)$-approximation to the sum of squared distances from the n rows of A to *any* set of k affine subspaces, each of dimension at most j. Our embedding yields $(0,k)$-coresets of size $\mathcal{O}(k)$ for handling k-means queries, $(j,1)$-coresets of size $\mathcal{O}(j)$ for PCA queries, and (j,k)-coresets of size $(\log n)^{\mathcal{O}(jk)}$ for any $j,k \geq 1$ and constant $\varepsilon \in (0,1/2)$. Previous coresets usually have a size which is linearly or even exponentially dependent of d, which makes them useless when $d \sim n$.

Using our coresets with the merge-and-reduce approach, we obtain embarrassingly parallel streaming algorithms for problems such as k-means, PCA and projective clustering. These algorithms use update time per point and memory that is polynomial in $\log n$ and only linear in d.

For cost functions other than squared Euclidean distances we suggest a simple recursive coreset construction that produces coresets of size $k^{1/\varepsilon^{\mathcal{O}(1)}}$ for k-means and a special class of bregman divergences that is less dependent on the properties of the squared Euclidean distance.

*MIT, Distributed Robotics Lab. Email: dannyf@csail.mit.edu

†TU Dortmund, Germany, Email: {melanie.schmidt, christian.sohler}@tu-dortmund.de

1 Introduction

Big Data. Scientists regularly encounter limitations due to large data sets in many areas. Data sets grow in size because they are increasingly being gathered by ubiquitous information-sensing mobile devices, aerial sensory technologies (remote sensing), genome sequencing, cameras, microphones, radio-frequency identification chips, finance (such as stocks) logs, internet search, and wireless sensor networks [30, 38]. The world's technological per-capita capacity to store information has roughly doubled every 40 months since the 1980s [31]; as of 2012, every day 2.5 etabytes(2.5×10^{18}) of data were created [4]. Data sets as the ones described above and the challenges involved when analyzing them is often subsumed in the term *Big Data*.

Gartner, and now much of the industry use the "3Vs" model for describing Big Data [14]: increasing *volume* n (amount of data), its *velocity* (update time per new observation) and its *variety* d (dimension, or range of sources). The main contribution of this paper is that it deals with cases where both n and d are huge, and does not assume $d \ll n$.

Data analysis. Classical techniques to analyze and/or summarize data sets include clustering, i.e. the partitioning of data into subsets of similar characteristics, and dimension reduction which allows to consider the dimensions of a data set that have the highest variance. In this paper we mainly consider problems that minimize the sum of squared error, i.e. we try to find a set of geometric centers (points, lines or subspaces), such that the sum of squared distances from every input point to its nearest center is minimized.

Examples are the *k-means* or sum of squares clustering problem where the centers are points. Another example is the *j-subspace mean* problem, where $k = 1$ and the center is a j-subspace, i. e., the sum of squared distances to the points is minimized over all j-subspaces. The *j-rank approximation* of a matrix is the projection of its rows on their j-subspace mean. *Principal component analysis (PCA)* is another example where $k = 1$, and the center is an

affine subspace. Constrained versions of this problem (that are usually NP-hard) include the non-negative matrix factorization (NNMF) [2], when the j-subspace should be spanned by positive vectors, and Latent Dirichlet Allocation (LDA) [3] which generalizes NNMF by assuming a more general prior distribution that defines the probability for every possible subspace.

The most general version of the problem that we study is the linear or affine j-Subspace k-Clustering problem, where the centers are j-dimensional linear or affine subspaces and $k \geq 1$ is arbitrary.

In the context of Big Data it is of high interest to find methods to reduce the size of the streaming data but keep its main characteristics according to these optimization problems.

Coresets. A small set of points that approximately maintains the properties of the original point with respect to a certain problem is called a *coreset* (or core-set). Coresets are a data reduction technique, which means that they tackle the first two 'Vs' of Big Data, volume n and velocity (update time), for a large family of problems in machine learning and statistics. Intuitively, a coreset is a semantic compression of a given data set. The approximation is with respect to a given (usually infinite) set Q of query shapes: for every shape in Q the sum of squared distances from the original data and the coreset is approximately the same. Running optimization algorithms on the small coreset instead of the original data allows us to compute the optimal query much faster, under different constraints and definition of optimality.

Coresets are usually of size at most logarithmic in the number n of observations, and have similar update time per point. However, they do not handle the variety of sources d in the sense that their size is linear or even exponential in d. In particular, existing coreset are useless for dealing with Big Data when $d \sim n$. In this paper we suggest coresets of size independent of d , while still independent or at most logarithmic in n.

Non-SQL databases. Big data is difficult to work with using relational databases of n records in d columns. Instead, in NOSQL is a broad class of database management systems identified by its non-adherence to the widely used relational database management system model. In NoSQL, every point in the input stream consists of tuples (object, feature, value), such as ("Scott", "age", 25). More generally, the tuple can be decoded as $(i, j, value)$ which means that the entry of the input matrix in the ith row and jth column is $value$. In particular, the total number of observations and dimensions (n and d) is unknown while passing over the streaming data. Our paper support this non-relational model in the sense that, unlike most of previous results, we do not assume that either d or n are bounded or known in advance. We only assume that the first coordinates (i, j) are increasing for every new inserted value.

2 Related work

Coresets. The term coreset was coined by Agarwal, Har-Peled and Varadarajan [10] in the context of extend measures of point sets. They proved that every point set P contains a small subset of points such that for any direction, the directional width of the point set will be approximated. They used their result to obtain kinetic and streaming algorithms to approximately maintain extend measures of point sets.

Application to Big Data. An off-line coreset construction can immediately turned into streaming and embarrassingly parallel algorithms that use small amount of memory and update time. This is done using a merge-and-reduce technique as explained in Section 10. This technique makes coresets a practical and provably accurate tool for handling Big data. The technique goes back to the work of Bentley and Saxe [11] and has been first applied to turn coreset constructions into streaming algorithms in [10]. Popular implementations of this technique include Hadoop [40].

Coresets for k-points clustering ($j = 0$). The first coreset construction for clustering problems was done by Badoiu, Har-Peled and Indyk [13], who showed that for k-center, k-median and k-means clustering, an approximate solution has a small witness set (a subset of the input points) that can be used to generate the solution. This way, they obtained improved clustering algorithms. Har-Peled and Mazumdar [29] gave a stronger definition of coresets for k-median and k-means clustering. Given a point set P this definition requires that for *any set* of k centers C the cost of the weighted coreset S with respect to P is approximated upto a factor of $(1 + \varepsilon)$. Here, S is not necessarily a subset of P. We refer to their definition as a *strong coreset*.

Har-Peled and Kushal [28] showed strong coresets for low-dimensional space, of size independent of the number of input points n. Frahling and Sohler designed a strong coreset that allows to efficiently maintain a coreset for k-means in dynamic data streams [24]. The first construction of a strong coreset for k-median and k-means of size polynomial in the dimension was done by Chen [15]. Langberg and Schulman [35] defined the notion of *sensitivity* of an

input point, and used it to construct strong coresets of size $O(k^2d^2/\varepsilon^2)$, i.e, independent of n. Feldman and Langberg [21] showed that small total sensitivity and VC-dimension for a family of shapes yields a small coreset, and provided strong coresets of size $O(kd/\varepsilon^2)$ for the k-median problem.

Feldman, Monemizadeh and Sohler [22] gave a construction of a weak coreset of size independent of both the number of input points n and the dimension d. The disadvantage of weak coresets is that, unlike the result in this paper, they only give a guarantee for centers coming from a certain set of candidates (instead of any set of centers as in the case of strong coresets).

Feldman and Schulman [23] developed a strong coreset for the k-median problem when the centers are weighted, and for generalized distance functions that handle outliers. There are also efficient implementation of k-means approximation algorithm based on coresets, both in the streaming [9] and non-streaming setting [25].

j-subspace clustering ($k = 1$). Coresets have also been developed for the subspace clustering problem. The j-dimensional subspace of $\mathbb{R}^d$ that minimizes the sum of squared distances to n input points can be computed in $O(\min\{nd^2, dn^2\})$ time using the Singular Value Decomposition (SVD). The affine j-dimensional subspace (i.e, flat that does not intersect the origin) that minimizes this sum can be computed similarly using principle component analysis (PCA), which is the SVD technique applied after translating the origin of the input points to their mean.

In both cases, the solution for the higher dimensional $(j + 1)$-(affine)-subspace clustering problem can be obtained by extending the solution to the j-(affine)-subspace clustering problem by one dimension. This implicitely defines an orthogonal basis of $\mathbb{R}^d$. The vectors of these bases are called *right singular vectors*, for the subspace case, and *principle components* for the affine case.

A line of research developed weak coresets for faster approximations [16, 17, 39, 26]. These results are usually based on the idea to sample sets of points with probability roughly proportional to the volume of the simplex spanned by the points, because such a simplex will typically have a large extension in the directions defined by the first left singular vectors. See recent survery in [8] for weak coresets.

Feldman, Fiat and Sharir [19] developed a strong coreset for the subspace approximation problem in low dimensional spaces, and Feldman, Monemizadeh, Sohler and Woodruff [20] provided such a coreset for high dimensional spaces that is polynomial in the dimension of the input space d and exponential in the considered subspace j. Feldman and Langberg [21] gave a strong coreset whose size is polynomial in both d and j.

Constrained optimization (e.g. NNMF, LDA). Non-negative matrix factorization (NNMF) [2], and Latent Dirichlet Allocation (LDA) [3] are constrained versions of the j-subspace problem. In NNMF the desired j-subspace should be spanned by positive vectors, and LDA is a generalization of NNMF, where we are given additinal prior probabilities (e.g., multiplicative cost) for every candidate solution. An important practical advantage of coresets (unlike weak coresets) is that they can be used with existing algorithms and heuristics for such constrained optimization problems. The reason is that strong coreset approximates the sum of squared distances to *any* given shape from a family of shapes (in this case, j-subspaces), independently for a specific optimization problem. In particular, running heuristics for a constrained optimization problem on an ε-coreset would yield the same quality of approximations compared to such run on the original input, up to provable $(1+\varepsilon)$-approximation. While NNMF and LDA are NP-hard, coresets for j-subspaces can be constructed in polynomial (and usually practical) time.

k-lines clustering ($j = 1$). For k-lines clustering, Feldman, Fiat and Sharir [19] show strong coresets for low-dimensional space, and [21] improve the result for high-dimensional space. In [23] the size of the coreset reduced to be polynomial in $2^{O(k)} \log n$. Har-Peled proved a lower bound of $\min\{2^k, \log n\}$ for the size of such coresets.

Projective clustering ($k, j > 1$). For general projective clustering, where $k, j \geq 1$, Har-Peled [27] showed that there is no strong coreset of size sublinear in n, even for the family of pair of planes in $\mathbb{R}^3$ ($j = k = 2, d = 3$). However, recently Varadarajan and Xiao [36] showed that there is such a coreset if the input points are on an integer grid whose side length is polynomial in n.

Bregman divergences. Bregman divergences are a class of distance measures that are used frequently in machine learning and they include l_2^2. Banerjee et al. [12] generalize the Lloyd's algorithm to Bregman divergences for clustering points ($j = 0$). In general, Bregman divergences have singularities at which the cost might go to infinity. Therefore, researchers studied clustering for so-called μ-similar Bregman divergences, where there is an upper and lower bound by constant μ times a Mahalanobis distance [7]. The only known coreset construction for clustering under Bregman divergences is a weak coreset of

size $O(k \log n/\varepsilon^2 \log(|\Gamma|^k \log n))$ by Ackermann and Blömer [6].

3 Background and Notation

We deal with clustering problems on point sets in Euclidean space, where a point set is represented by the rows of a matrix A. The input matrix A and all other matrices in this paper are over the reals.

Notations and Assumptions. The number of input points is denoted by n. For simplicity of notation, we assume that $d = n$ and thus A is an $n \times n$ matrix. Otherwise, we add $(n-d)$ columns (or $d-n$ rows) containing all zeros to A. We label the entries of A by $a_{i,j}$. The ith row of A will be denoted as A_{i*} and the jth columns as A_{*j}.

The identity matrix of $\mathbb{R}^j$ is denoted as $I \in \mathbb{R}^{j \times j}$. For a matrix X with entries $x_{i,j}$, we denote the *Frobenius norm* of X by $\|X\|_2 = \sqrt{\sum_{i,j} x_{i,j}^2}$. We say that a matrix $X \in \mathbb{R}^{n \times j}$ has *orthonormal columns* if its columns are orthogonal unit vectors. Such a matrix is also called *orthogonal* matrix. Notice that every orthogonal matrix X satisfies $X^T X = I$.

The columns of a matrix X span a linear subspace L if all points in L can be written as linear combinations of the columns of A. This implies that the columns contain a basis of L.

A j-dimensional linear subspace $L \subseteq \mathbb{R}^n$ will be represented by an $n \times j$ basis matrix X with orthonormal columns that span L. The projection of a point set (matrix) A on a linear subspace L represented by X will be the matrix (point set) $AX \in \mathbb{R}^{n \times j}$. These coordinates are with respect to the column space of X. The projections of A on L using the coordinates of $\mathbb{R}^n$ are the rows of AXX^T.

Distances to Subspaces. We will often compute the squared Euclidean distance of a point set given as a matrix A to a linear subspace L. This distance is given by $\|AY\|_2^2$, where Y is an $n \times (n-j)$ matrix with orthonormal columns that span $L^\perp$, the orthogonal complement of L. Therefore, we will also sometimes represent a linear subspace L by such a matrix Y.

An affine subspace is a translation of a linear subspace and as such can be written as $p + L$, where $p \in \mathbb{R}^n$ is the translation vector and L is a linear subspace.

For a compact set $S \subseteq \mathbb{R}^d$ and a vector p in $\mathbb{R}^d$, we denote the Euclidean distance between p and (its closest points in) S by

$$\text{dist}^2(p, S) := \min_{s \in S} \|p - s\|_2^2.$$

For an $n \times d$ matrix A whose rows are $p_1, \cdots, p_n$, we define the sum of the squared distances from A to S by

$$\text{dist}^2(A, S) = \sum_{i=1}^{n} \text{dist}^2(p_i, S). \tag{3.1}$$

Thus, if L is a linear j-subspace and Y is a matrix with $n-j$ orthonormal columns spanning $L^\perp$, then $\text{dist}(p, L) = \|p^T Y\|_2$. We generalize the notation to $n \times n$ matrices and define, $\text{dist}(A, L) = \sum_{i=1}^n \text{dist}(A_{i*}, L)$ for any compact set L. Here A_{i*} is the ith row of A. Furthermore, we write $\text{dist}^2(A, L) = \sum_{i=1}^n \big(\text{dist}(A_{i*}, L)\big)^2$.

DEFINITION 1. (Linear (Affine) j-Subspace k-Clustering) *Given a set of n points in n-dimensional space as an $n \times n$ matrix A, the k j-subspace clustering problem is to find a set L of k linear (affine) j-dimensional subspaces $L_1, \ldots, L_k$ of $\mathbb{R}^d$ that minimizes the sum of squared distances to the nearest subspace, i.e.,*

$$\text{cost}(A, L) = \sum_{i=1}^{n} \min_{j=1\ldots,k} \text{dist}^2(A_{i*}, L_j)$$

is minimized over every choice of $L_1, \cdots, L_k$.

Singular Value Decomposition. An important tool from linear algebra that we will use in this paper is the singular value decomposition of a matrix A. Recall that $A = UDV^T$ is the Singular Value Decomposition (SVD) of A if $U, V \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with non-increasing entries. Let $(s_1, \cdots, s_n)$ denote the diagonal of D^2.

For an integer j between 0 to n, the first j columns of V span a linear subspace L^* that minimize the sum of squared distances to the points (rows) of A, over all j-dimensional linear subspaces in $\mathbb{R}^n$. This sum equals $s_{j+1} + \ldots + s_n$, i.e. for any $n \times (n-j)$ matrix Y with orthonormal columns, we have

$$\|AY\|_2^2 \geq \sum_{i=j+1}^{n} s_i. \tag{3.2}$$

The projection of the points of A on L^* are the rows of UD. The sum of the squared projection of the points of A on L^* is $s_1 + \ldots + s_j$. Note that the sum of squared distances to the origin (the optimal and only 0-subspace) is $s_1 + \ldots + s_n$.

Coresets. In this paper we introduce a new notion of coresets, which is a small modification of the earlier definition by Har-Peled and Mazumdar [29] for the k-median and k-means problem (here adapted to the more general setting of subspace clustering)

that is commonly used in coreset constructions for these problems. The new idea is to allow to add a constant C to the cost of the coreset. Interestingly, this simple modification allows us to obtain improved coreset constructions.

DEFINITION 2. *Let A be an $n \times d$ matrix whose rows represents n points in $\mathbb{R}^d$. An $m \times n$ matrix M is called (k,ε)-coreset for the j-subspace k-clustering problem of A, if there is a constant c such that for every choice of k j-dimensional subspaces $L_1,\ldots,L_k$ we have*

$$(1-\varepsilon)\operatorname{cost}(A,L) \leq \operatorname{cost}(M,L)+c \leq (1+\varepsilon)\operatorname{cost}(A,L).$$

4 Our results

In this section we summarize our results.

The main technical result is a proof that the sum of squared distances from a set of points in $\mathbb{R}^d$ (rows of an $n \times d$ matrix A) to any other compact set that is spanned by k vectors of $\mathbb{R}^d$ can be $(1+\varepsilon)$-approximated using the $O(k/\varepsilon)$-rank approximation of A, together with a constant that depends only on A. Here, a distance between a point p to a set is the Euclidean distance of p to the closest point in this set. The $O(k/\varepsilon)$-rank approximation of A is the projection of the rows of A on the k-dimensional subspace that minimizes their sum of squared distances. Hence, we prove that the low rank approximation of A can be considered as a coreset for its n rows. While the coreset also has n rows, its dimensionality is independent of d, but only on k and the desired error. Formally:

THEOREM 4.1. *Let A be an $n \times d$ matrix, $k \geq 1$ be an integer and $0 < \varepsilon < 1$. Suppose that A_m is the m-rank approximation of A, where $m := b\lceil k/\varepsilon^2 \leq n$ for a sufficiently large constant b. Then for every compact set S that is contained in a k-dimensional subspace of $\mathbb{R}^d$, we have*

$$\begin{aligned}&(1-\varepsilon)\operatorname{dist}^2(A,S) \leq \operatorname{dist}^2(A_m,S) + \|A-A_m\|^2\\ \leq&(1+\varepsilon)\operatorname{dist}^2(A,S),\end{aligned}$$

where $\operatorname{dist}^2(A,S)$ is the sum of squared distances from each row on A to its closest point in S.

Note that A takes nd space, while the pair A_m and the constant $\|A-A_m\|$ can be stored using $nm+1$ space.

Coresets. Combining our main theorem with known results [21, 19, 36] we gain the following coresets for projective clustering. All the coresets can handle Big data (streaming, parallel computation and fast update time), as explained in Section 10 and later in this section.

COROLLARY 4.1. *Let P be a set of points in $\mathbb{R}^d$, and $k,j \geq 0$ be a pair of integers. There is a set Q in $\mathbb{R}^d$, a weight function $w : Q \to [0,\infty)$ and a constant $c > 0$ such that the following holds.*

For every set B which is the union of k affine j-subspaces of $\mathbb{R}^d$ we have

$$\begin{aligned}&(1-\varepsilon)\sum_{p\in P}\operatorname{dist}^2(p,B) \leq \sum_{p\in Q} w(p)\operatorname{dist}^2(p,B)+c\\ \leq&(1+\varepsilon)\sum_{p\in P}\operatorname{dist}^2(p,B),\end{aligned}$$

and

1. $|Q| = O(j/\varepsilon)$ *if* $k = 1$
2. $|Q| = O(k^2/\varepsilon^4)$ *if* $j = 0$
3. $|Q| = \operatorname{poly}(2^k \log n, 1/\varepsilon)$ *if* $j = 1$,
4. $|Q| = \operatorname{poly}(2^{kj}, 1/\varepsilon)$ *if* $j,k > 1$, *under the assumption that the coordinates of the points in P are integers between 1 and $n^{O(1)}$.*

In particular the size of Q is independent of d.

PCA and k-rank approximation. For the first case of the last theorem, we obtain a small coreset for k-dimensional subspaces with no multiplicative weights, which contains only $O(k/\varepsilon)$ points in $\mathbb{R}^d$. That is, its size is independent of both d and n.

COROLLARY 4.2. *Let A be an $n \times d$ matrix. Let $m = \lceil k/\varepsilon \rceil + k - 1$ for some $k \geq 1$ and $0 < \varepsilon < 1$ and suppose that $m \leq n-1$. Then, there is an $m \times d$ matrix $\tilde{A}$ and a constant $c \geq 1$, such that for every k-subspace S of $\mathbb{R}^d$ we have*

$$\begin{aligned}&(1-\varepsilon)\operatorname{dist}^2(A,S) \leq \operatorname{dist}^2(\tilde{A},S)+c\\ \leq&(1+\varepsilon)\operatorname{dist}^2(A,S)\end{aligned} \tag{4.3}$$

Equality (4.3) can also be written using matrix notation: for every $d \times (d-k)$ matrix Y whose columns are orthonormal we have

$$\begin{aligned}&(1-\varepsilon)\|AY\|^2 \leq \|\tilde{A}Y\|+c\\ \leq&(1+\varepsilon)\|AY\|^2.\end{aligned}$$

Notice that in Theorem 4.1, A_m is still n-dimensional and holds n points. We obtain Corollary 4.2 by obbserving that $\|A_m X\| = \|D^{(m)}V^T X\|$ for every $X \in \mathbb{R}^{n\times n-j}$. As the only non-zero entries of $D^{(m)}V^T$ are in the first m rows, we can store these as the matrix $\tilde{A}$. This can be considered as an exact coreset for A_m.

Streaming. In the streaming model, the input points (rows of A) arrive on-line (one by one) and

we need to maintain the desired output for the points that arrived so far. We aim that both the update time per point and the required memory (space) will be small. Usually, linear in d and polynomial in $\log n$.

For computing the k-rank approximation of a matrix A efficiently, in parallel or in the streaming model we cannot use Corollary 4.2 directly: First, because it assumes that we already have the $O(k/\varepsilon)$-rank approximation of A, and second, that A assumed to be in memory which takes nd space. However, using merge-and-reduce we only need to apply the construction of the theorem on very small matrices A of size independent of d in overall time that is linear in both n and d, and space that is logarithmic in $O(\log n)$. The construction is also embarrassingly parallel; see Fig. 3 and discussion in Section 10.

The following corollary follows from Theorem 10.1 and the fact that computing the SVD for an $m \times d$ matrix takes $O(dm^2)$ time when $m \leq d$.

COROLLARY 4.3. *Let A be the $n \times d$ matrix whose n rows are vectors seen so far in a stream of vectors in $\mathbb{R}^d$. For every $n \geq 1$ we can maintain a matrix $\tilde{A}$ and $c \geq 0$ that satisfies (4.3) where*

1. *$\tilde{A}$ is of size $2m \times 2m$ for $m = \lceil k/\varepsilon \rceil$*
2. *The update time per row insertion, and overall space used is*

$$d \cdot \left(\frac{k \log n}{\varepsilon} \right)^{O(1)}$$

Using the last corollary, we can efficiently compute a $(1+\varepsilon)$-approximation to the subspace that minimizes the sum of squared distances to the rows of a huge matrix A. After computing $\tilde{A}$ and c for A as described in Corollary 4.3, we compute the k-subspace S^* that minimizes the sum of squared distances to the small matrix $\tilde{A}$. By (4.3), S^* approximately minimizes the sum of squared distances to the rows of A. To obtain an approximation to the k-rank approximation of A, we project the rows of A on S^* in $O(ndk)$ time.

Since $\tilde{A}$ approximates $\text{dist}^2(A, S)$ for *any* k-subspace of $\mathbb{R}^d$ (not only S^*), computing the subspace that minimizes $\text{dist}^2(\tilde{A}, S)$ under arbitrary constraints would yield a $(1+\varepsilon)$-approximation to the subspace that minimizes $\text{dist}^2(A, S)$ under the same constraints. Such problems include the non-negative matrix factorization (NNMF, also called pLSA, or probabilistic LSA) which aims to compute a k-subspace S^* that minimizes sum of squared distances to the rows of A as defined above, with the additional constraint that the entries of S^* will all be non-negative.

Latent Drichlet analysis (LDA) [3] is a generalization of NNMF, where a prior (multiplicative weight) is given for every possible k-subspace in $\mathbb{R}^d$. In practice, especially when the corresponding optimization problem is NP-hard (as in the case of NNMF and LDA), running popular heuristics on the coreset pair $\tilde{A}$ and c may not only turn them into faster, streaming and parallel algorithms. It might actually yield *better* results (i.e, "$1-\varepsilon$" approximations) compared to running the heuristics on A; see [33].

In principle component analysis (PCA) we usually interested in the *affine* k-subspace that minimizes the sum of squared distances to the rows of A. That is, the subspace may not intersect the origin. To this end, we replace k by $k+1$ in the previous theorems, and compute the optimal affine k-subspace rather than the $(k+1)$ optimal subspace of the small matrix $\tilde{A}$.

For the k-rank approximation we use the following corollary with Z as the empty set of constraints. Otherwise, for PCA, NNMF, or LDA we use the corresponding constraints.

COROLLARY 4.4. *Let A be an $n \times d$ matrix. Let A_k denote an $n \times k$ matrix of rank at most k that minimizes $\|A - A_k\|^2$ among a given (possibly infinite) set Z of such matrices. Let $\tilde{A}$ and c be defined as in Corollary 4.3, and let $\tilde{A}_k$ denote the matrix that minimizes $\|\tilde{A} - \tilde{A}_k\|^2$ among the matrices in Z. Then*

$$\|A - \tilde{A}_k\|^2 \leq (1+\varepsilon)\|A - A_k\|^2.$$

Moreover, $\|A - A_k\|^2$ can be approximated using $\tilde{A}$ and c, as

$$\begin{aligned}(1-\varepsilon)\|A - \tilde{A}_k\|^2 &\leq \|\tilde{A} - \tilde{A}_k\|^2 + c \\ &\leq (1+\varepsilon)\|A - \tilde{A}_k\|^2.\end{aligned}$$

k-means. The k-mean of A is the set S^* of k *points* in $\mathbb{R}^d$ that minimizes the sum of squared distances $\text{dist}^2(A, S^*)$ to the rows of A among every k points in $\mathbb{R}^d$. It is not hard to prove that the k-mean of the k-rank approximation A_k of A is a 2-approximation for the k-mean of A in term of sum of squared distances to the k centers [18]. Since every set of k points is contained in a k-subspace of $\mathbb{R}^d$, the k-mean of A_m is a $(1+\varepsilon)$-approximation to the k-means of A in term of sum of squared distances. Since the k-mean of $\tilde{A}$ is clearly in the span of $\tilde{A}$, we conclude from our main theorem the following corollary that generalizes the known results from 2-approximation to $(1+\varepsilon)$-approximation.

COROLLARY 4.5. *Let A_m denote the $m = bk/\varepsilon^2$ rank approximation of an $n \times d$ matrix A, where b is*

a sufficiently large constant. Then the sum of the squared distances from the rows of A to the k-mean of A_m is a $(1+\varepsilon)$-approximation for the sum of squared distances to the k-mean of A.

While the last corollary projects the input points to a $O(k/\varepsilon)$-dimensional subspace, the number of rows (points) is still n. We use existing coreset constructions for k-means on the lower dimensional points. These constructions are independent of the number of points and linear in the dimension. Since we apply these coresets on A_m, the resulting coreset size is independent of both n and d.

Notice the following 'coreset' of a similar type for the case $k = 1$. Let $\overline{A}$ be the mean of A. Then the following 'triangle inequality' holds for every point $s \in \mathbb{R}^d$:

$$\text{dist}^2(A, s) = \text{dist}^2(A, \{\overline{A}\}) + n \cdot \text{dist}^2(\overline{A}, s).$$

Thus, $\overline{A}$ forms an exact coreset consisting of one d-dimensional point together with the constant $\text{dist}^2(A, \{\overline{A}\})$.

As in our result, and unlike previous coreset constructions, this coreset for 1-mean is of size independent of d and uses additive constant on the right hand side. Our results generalizes this exact simple coresets for k-means where $k \geq 2$ while introducing $(1+\varepsilon)$-approximation.

Unlike the k-rank approximation that can be computed using the SVD of A, the k-means problem is NP-hard when k is not a constant, analog to constrained versions (e.g. NNMF or LDA) that are also NP-hard. Again, we can run (possibly inefficient) heuristics or constant factor approximations for computing the k-mean of A under different constraints in the streaming and parallel model by running the corresponding algorithms on $\tilde{A}$.

Comparison to Johnson-Lindestrauss Lemma. The JL-Lemma states that projecting the n rows of an $n \times d$ matrix A on a random subspace of dimension $\Omega(\log(n)/\varepsilon^2)$ in $\mathbb{R}^d$ preserves the Euclidean distance between every pair of the rows up to a factor of $(1+\varepsilon)$, with high (arbitrary small constant) probability $1-\delta$. In particular, the k-mean of the projected rows minimizes the k-means cost of the original rows up to a factor of $(1+\varepsilon)$. This is because the minimal cost depends only on the $\binom{n}{2}$ distances between the n rows [34]. Note that we get the same approximation by the projection A_m of A on an m-dimensional subspace.

While our embedding also projects the rows onto a linear subspace of small dimension, its construction and properties are different from a random subspace. First, our embedding is deterministic (succeeds with probability 1, no dependency of time and space on δ). Second, the dimension of our subspace is $\lceil k/\varepsilon \rceil$, that is, independent of n and only linear in $1/\varepsilon$. Finally, the *sum* of squared distances to *any* set that is spanned by k-vectors is approximated, rather than the inter distance between each pair of input rows.

Generalizations for Bregman divergences. Our last result is a simple recursive coreset construction that yields the first strong coreset for k-clustering with μ-similar Bregman divergences. The coreset we obtain has size $k^{O(1/\varepsilon)}$. The idea behind the construction is to distinguish between the cases that the input point set can be clustered into k clusters at a cost of at most $(1-\varepsilon)$ times the cost of a 1-clustering or not. We apply the former case recursively on the clusters of an optimal (or approximate) solution until either the cost of the clustering has dropped to at most ε times the cost of an optimal solution for the input point set or we reach the other case. In both cases we can then replace the clustering by a so-called clustering feature containing the number of points, the mean and the sum of distances to the mean. Note that our result uses a stronger assumption than [6] because we assume that the divergence is μ-similar on the complete $\mathbb{R}^d$ while in [6] this only needs to hold for a subset $X \subset \mathbb{R}^d$. However, compared to [6] we gain a strong coreset, and the coreset size is independent of the number of points.

5 Implementation in Matlab

Our main coreset construction is easy to implement if a subroutine for SVD is already available, for example, Algorithm 1 shows a very short Matlab implementation. The first part initializes the $n \times d$-matrix A with random entries, and also sets the parameters j and ε. In the second part, the actual coreset construction happens. We set $m := \lceil j/\varepsilon \rceil + j - 1$ according to Corollary 6.1. Then we calculate the first m singular vectors of A and store it in the matrices U, D and V. Notice that for small matrices, it is better to use the full svd *svd(A,0)*. Then, we compute the coreset matrix C and the constant $c = \sum_{i=m+1}^{n} s_i$. In the third part, we check the quality of our coreset. For this, we compute a random query subspace represented by a matrix Q with j random orthogonal columns, calculate the sum of the squared distances of A to Q, and of C to Q. In the end, *err* contains the multiplicative error of our coreset.

The Matlab code of Algorithm 2 is based on Corollary 4.5 for constructing a coreset C for k-mean queries. This coreset has n points that are lying on $O(k/\varepsilon^2)$ subspace, rather than the original d-dimensional space. Note that further reduction for

a coreset of size independent of both n and d can be obtained by applying additional construction on C; see Corollary 9.1. The code in Algorithms 1 and 2 is only for demonstration and explanation purposes. In particular, it should be noted that

- For simplicity, we choose A as a random Gaussian matrix. Real data sets usually contain more structure.
- We choose the value of m according to the corresponding theorems, that are based on worst case analysis. In practice, as can be seen in our experiments, significantly smaller values for m can be used to obtain the same error bound. The desired value of m can be chosen using hill climbing or binary search techniques on the integer m.
- In Algorithm 1 we compare the coreset approximation for a fixed query subspace, and not the maximum error over all such subspaces, which is bounded in Corollary 6.1.
- In Algorithm 2 we actually get a *better* solution using the coreset, compared to the original set ($\varepsilon < 0$). This is possible since we used Matlab's heuristic for k-means and not an optimal solution.

```
>> %% Coreset Computation                %%%
>> %%         for j-Subspace Approximation %%%

>> % 1. Creating Random Input
>> n=10000;
>> d=2000;
>> A=rand(n,d);
>> j=2;
>> epsilon=0.1;

>> % 2. The Actual Coreset Construction
>> m=j+ceil(j/epsilon)-1;  % i.e., m=21.
>> [U, D, V]=svds(A,m);
>> C = D*V'; % C is an m-by-m matrix
>> c=norm(A,'fro')^2-norm(D,'fro')^2;

>> % 3. Compare sum of squared distances of j
>> % random orthogonal columns Q to A and C
>> Q=orth(rand(d,d-j));
>> costA= norm(A*Q,'fro')^2;
>> costC= c+norm(C*Q,'fro')^2;
>> errJsubspaceApprox = abs(costC/costA - 1)

errJsubspaceApprox =  2.4301e-004
```

Algorithm 1: A Matlab implementation of the coreset construction for j-subspace approximation.

6 Subspace Approximation for one Linear Subspace

We will first develop a coreset for the problem of approximating the sum of squared distances of a point set to *one* linear j-dimensional subspace. Let $L \subseteq \mathbb{R}^n$ be a j-dimensional subspace represented by an $n \times j$ matrix X with orthonormal columns spanning X and an $n \times (n-j)$ matrix Y with orthonormal columns spanning the orthogonal complement $L^\perp$ of L. Above, we recalled the fact that for a given matrix $A \in \mathbb{R}^{n \times n}$ containing n points of dimension n as its rows, the sum of squared distances $\|AY\|_2^2$ of the points to L is at least $\sum_{i=j+1}^{n} s_i$, where s_i is the ith singular value of A (sorted non increasingly). When L is the span of the first j eigenvectors of an orthonormal basis of A^TA (where the eigen vectors [=left singular vectors of A] are sorted according to the corresponding left singular values), then this value is exactly matched. Thus, the subspace that is spanned by the first j left singular vectors of A achieves a minimum value of $\sum_{i=j+1}^{n} s_i$.

Now, we will show that $m := j + \lceil j/\varepsilon \rceil - 1$ appropriately chosen points suffice to approximately compute the cost of *every* j-dimensional subspace L. We obtain these points by considering the singular value decomposition $A = UDV^T$. Our first step is to replace the matrix A by its best rank m approximation with respect to the squared Frobenius norm, namely, by $UD^{(m)}V^T$, where $D^{(m)}$ is the matrix that contains the m largest diagonal entries of D and that is 0 otherwise. We show the following simple lemma about the error of this approximation with respect to squared Frobenius norm.

LEMMA 6.1. *Let $A \in \mathbb{R}^{n \times n}$ be an $n \times n$ matrix with the singular value decomposition $A = UDV^T$, and let X be a $n \times j$ matrix whose columns are orthonormal. Let $\varepsilon \in (0, 1]$ and $m \in \mathbb{N}$ with $n - 1 \geq m \geq j + \lceil j/\varepsilon \rceil - 1$, and let $D^{(m)}$ be the matrix that contains the first m diagonal entries of D and is 0 otherwise. Then*

$$0 \leq \|UDV^TX\|_2^2 - \|UD^{(m)}V^TX\|_2^2 \leq \varepsilon \cdot \sum_{i=j+1}^{n} s_i.$$

Proof. We first observe that $\|UDV^TX\|_2^2 -$

```
>> %% Coreset Computation for 2-means    %%%

>> % 1. Creating Random Input
>> n=10000;
>> d=2000;
>> A=rand(n,d); % n-by-d random matrix
>> j=2;
>> epsilon=0.1;

>> %2. Coreset Construction
>> m=ceil(j/epsilon^2)+j-1; % i.e, m=201
>> [U, D, V]=svds(A,m);
>> c=norm(A,'fro')^2-norm(D,'fro')^2;
>> C = U*D*V; % C is an n-by-m matrix

>> % 3. Compute k-mean for A and its coreset
>> k=2;
>> [~,centersA]=...
       kmeans(A,k,'onlinephase','off');
>> [~,centersC]=...
       kmeans(C,k,'onlinephase','off');

>> % 4.Evaluate sum of squared distances
>> [~,distsAA]=knnsearch(centersA,A);
>> [~,distsCA]=knnsearch(centersC,A);
>> [~,distsCC]=knnsearch(centersC,C);
>> costAA=sum(distsAA.^2); % opt. k-means
>> costCA=sum(distsCA.^2); % appr. cost
>> costCC=sum(distsCC.^2);

>> % Evaluate quality of coreset solution
>> epsApproximation = costCA/costAA - 1

epsApproximation = -4.2727e-007

>> % Evaluate quality of cost estimation
>> % using coreset
>> epsEstimation = abs(costCA/(costCC+c) - 1)

epsEstimation =  5.1958e-014
```

Algorithm 2: A Matlab implementation of the coreset construction for k-means queries.

$\|UD^{(m)}V^TX\|_2^2$ is always non-negative. Then

$$\begin{aligned} & \|UDV^TX\|_2^2 - \|UD^{(m)}V^TX\|_2^2 \\ = \; & \|DV^TX\|_2^2 - \|D^{(m)}V^TX\|_2^2 \\ = \; & \|(D-D^{(m)})V^TX\|_2^2 \\ \leq \; & \|(D-D^{(m)})\|_2^2\|X\|_S^2 \\ = \; & j \cdot s_{m+1} \end{aligned}$$

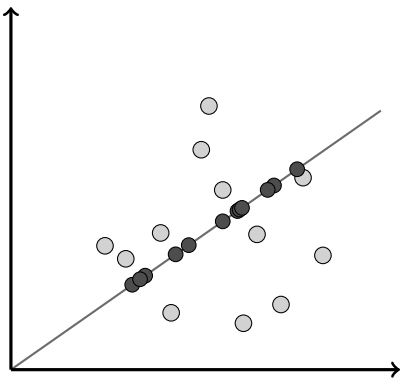

Figure 1: Vizualization of a point set that is projected down to a 1-dimensional subspace. Notice that both subspace here and in all other pictures are of the same dimension to keep the picture 2-dimensional, but the query subspace should have smaller dimension.

where the first equality follows since U has orthonormal columns, the second inequality since for $M = V^TX$ we have $\|DM\|_2^2 - \|D^{(m)}M\|_2^2 = \sum_{i=1}^n \sum_{j=1}^n s_i m_{ij}^2 - \sum_{i=1}^m \sum_{j=1}^n s_i m_{ij}^2 = \sum_{i=m+1}^n \sum_{j=1}^n s_i m_{ij}^2 = \|(D-D^{(m)})M\|^2$, and the inequality follows because the spectral norm is consistent with the Euclidean norm. It follows for our choice of m that

$$js_{m+1} \leq \varepsilon \cdot (m-j+1)s_{m+1} \leq \varepsilon \cdot \sum_{i=j+1}^{m+1} s_i \leq \varepsilon \cdot \sum_{i=j+1}^{n} s_i.$$

□

In the following we will show that one can use the first m rows of $D^{(m)}V^T$ as a coreset for the linear j-dimensional subspace 1-clustering problem. We exploit the observation that by the Pythagorean theorem it holds that $\|AY\|_2^2 + \|AX\|_2^2 = \|A\|_2^2$. Thus, $\|AY\|_2^2$ can be decomposed as the difference between $\|A\|_2^2$, the squared lengths of the points in A, and $\|AX\|_2^2$, the squared lengths of the projection of A on L. Now, when using $UD^{(m)}V^T$ instead of A, $\|UD^{(m)}V^TY\|_2^2$ decomposes into $\|UD^{(m)}V^T\|_2^2 - \|UD^{(m)}V^TX\|_2^2$ in the same way. By noting that $\|A\|_2^2$ is actually $\sum_{i=1}^n s_i$ and $\|UD^{(m)}V^T\|_2^2$ is $\sum_{i=1}^m s_i$, it is clear that storing the remaining terms of the sum is sufficient to account for the difference between these two norms. Approximating $\|AY\|_2^2$ by $\|UD^{(m)}V^TY\|_2^2$ thus reduces to approximate $\|AX\|_2^2$ by $\|UD^{(m)}V^TX\|_2^2$. We show that the difference between these two is only an ε-fraction of $\|AY\|_2^2$, and explain after the corollary why this implies the desired coreset result.

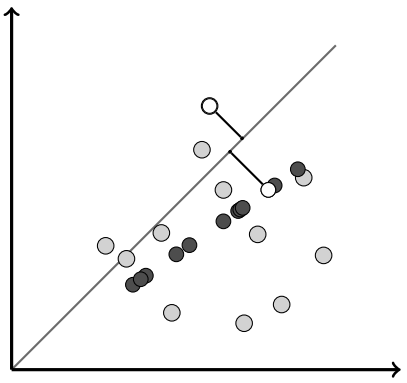

Figure 2: The distances of a point and its projection to a query subspace.

LEMMA 6.2. (Coreset for Linear Subspace 1-Clustering) *Let $A \in \mathbb{R}^{n\times n}$ be an $n \times n$ matrix with singular value decomposition $A = UDV^T$, Y be an $n \times (n-j)$ matrix with orthonormal columns, $0 \le \varepsilon \le 1$, and $m \in \mathbb{N}$ with $n-1 \ge m \ge j+\lceil j/\varepsilon \rceil - 1$. Then*

$$0 \le \|UD^{(m)}V^TY\|_2^2 + \sum_{i=m+1}^{n} s_i - \|AY\|_2^2 \le \varepsilon \cdot \|AY\|_2^2.$$

Proof. We have $\|AY\|_2^2 \le \|UD^{(m)}V^TY\|_2^2 + \|U(D-D^{(m)})V^TY\|_2^2 \le \|UD^{(m)}V^TY\|_2^2 + \sum_{i=m+1}^{n} s_i$, which proves that $\|UD^{(m)}V^TY\|_2^2 + \sum_{i=m+1}^{n} s_i - \|AY\|_2^2$ is non-negative. We now follow the outline sketched above. By the Pythagorean Theorem, $\|AY\|_2^2 = \|A\|_2^2 - \|AX\|_2^2$, where X has orthonormal columns and spans the space orthogonal to the space spanned by Y. Using, $\|A\|_2^2 = \sum_{i=1}^{n} s_i$ and, $\|UD^{(m)}V^T\|_2^2 = \sum_{i=1}^{m} s_i$, we obtain

$$\begin{aligned}
& \|UD^{(m)}V^TY\|_2^2 + \sum_{i=m+1}^{n} s_i - \|AY\|_2^2 \\
= \;& \|UD^{(m)}V^T\|_2^2 - \|UD^{(m)}V^TX\|_2^2 \\
& \quad + \sum_{i=m+1}^{n} s_i - \|A\|_2^2 + \|AX\|_2^2 \\
= \;& \|AX\|_2^2 - \|UD^{(m)}V^TX\|_2^2 \\
\le \;& \varepsilon \cdot \sum_{i=j+1}^{n} s_i \le \varepsilon \cdot \|AY\|_2^2
\end{aligned}$$

where the first inequality follows from Lemma 6.1. □

Now we observe that by orthonormality of the columns of U we have $\|UM\|_2^2 = \|M\|_2^2$ for any matrix M, which implies that $\|UDV^TX\|_2^2 = \|DV^TX\|_2^2$. Thus, we can replace the matrix $UD^{(m)}V^T$ in the above corollary by $D^{(m)}V^T$. This is interesting, because all rows except the first m rows of this new matrix have only 0 entries and so they don't contribute to $\|D^{(m)}V^TX\|_2^2$. Therefore, we will define our coreset matrix S to be matrix consisting of the first $m = O(j/\varepsilon)$ rows of $D^{(m)}V^T$. The rows of this matrix will be the coreset points.

In the following, we summarize our results and state them for n points in d-dimensional space.

COROLLARY 6.1. *Let A be an $n \times n$ matrix whose n rows represent n points in n-dimensional space. Let $A = UDV^T$ be the SVD of A and let $D^{(m)}$ be a matrix that contains the first $m = O(j/\varepsilon)$ diagonal entries of D and is 0 otherwise. Then the rows of $D^{(m)}V^T$ form a coreset for the linear j-subspace 1-clustering problem.*

If one is familiar with the coreset literature it may seem a bit strange that the resulting point set is unweighted, i.e. we replace n unweighted points by m unweighted points. However, for this problem the weighting can be implicitly done by scaling. Alternatively, we could also define our coreset to be the set of the first m rows of V^T where the ith row is weighted by s_i.

7 Dimensionality Reduction for Projective Clustering Problems

In order to deal with k subspaces we will use a form of dimensionality reduction. To define this reduction, let L be a linear j-dimensional subspace represented by an $n \times j$ matrix X with orthonormal columns and with Y being an $n \times (n-j)$ matrix with orthonormal columns that spans $L^{\perp}$. Notice that for any matrix M we can write the projection of the points in the rows of M to L as MXX^T, and that these projected points are still n-dimensional, but lie within the j-dimensional subspace L. Our first step will be to show that if we project both UDV^T and $A' := UD^{(m)}V^T$ on X by computing UDV^TXX^T and $UD^{(m)}V^TXX^T$, then the sum of squared distances of the corresponding rows of the projection is small compared to the cost of the projection. In other words, after the projection the points of A will be relatively close to their counterparts of A'. Notice the difference to the similar looking Lemma 6.1: In Lemma 6.1, we showed that if we project A to L and sum up the squared *lengths* of the projections, then this sum is similar to the sum of the lengths of the projections of A'. In the following corollary, we look at the distances between a projection of a point from A and the projection of the corresponding point in A', then we square these distances and show that the

sum of them is small.

COROLLARY 7.1. *Let $A \in \mathbb{R}^{n\times n}$ and let $A = UDV^T$ be its singular value decomposition. Let $D^{(m)}$ be a matrix whose first m diagonal entries are the same as that of D and let it be 0, otherwise. Let $X \in \mathbb{R}^{n\times j}$ be a matrix with orthonormal columns and let $n-1 \geq m \geq j + \lceil j/\varepsilon \rceil - 1$ and let $Y \in \mathbb{R}^{n\times(n-j)}$ a matrix with orthonormal columns that spans the orthogonal complement of the column space of X. Then*

$$\|UDV^TXX^T - UD^{(m)}V^TXX^T\|_2^2 \leq \varepsilon \cdot \|AY\|_2^2$$

Proof. We have $\|UDV^TXX^T - UD^{(m)}V^TXX^T\|_2^2 = \|DV^TX - D^{(m)}V^TX\|_2^2 = \|DV^TX\|_2^2 - \|D^{(m)}V^TX\|_2^2 = \|UDV^TX\|_2^2 - \|UD^{(m)}V^TX\|_2^2 \leq \varepsilon\sum_{i=j+1}^n s_i \leq \varepsilon\sum_{i=1}^n s_i \leq \varepsilon\|AY\|_2^2$, where the third last inequality follows from Lemma 6.1. □

Now assume we want to use $A' = UD^{(m)}V^T$ to estimate the cost of an j-dimensional affine subspace k-clustering problem. Let $L_1, \ldots, L_k$ be a set of affine subspaces and let L be a j^*-dimensional subspace containing $C = L_1 \cup \cdots \cup L_k$, $j^* \geq k(j+1)$. Then by the Pythagorean theorem we can write $\text{dist}^2(A,C) = \text{dist}^2(A,L) + \text{dist}^2(AXX^T, C)$, where X is a matrix with orthonormal columns whose span is L. We claim that $\text{dist}^2(A',C) + \sum_{i=m+1}^n s_i$ is a good approximation for $\text{dist}^2(A,C)$. We also know that $\text{dist}^2(A',C) = \text{dist}^2(A',L) + \text{dist}^2(A'XX^T, C)$. Furthermore, by Corollary 6.2, $|\text{dist}^2(A',L) + \sum_{i=m+1}^n s_i - \text{dist}^2(A,L)| \leq \varepsilon\text{dist}^2(A,L) \leq \varepsilon\text{dist}^2(A,C)$. Thus, if we can prove that $|\text{dist}^2(AXX^T,C) - \text{dist}^2(A'XX^T)|_2^2 \leq \varepsilon \cdot \text{dist}(A,C)$ we have shown that $|\text{dist}(A,C) - (\text{dist}(A',C) + \sum_{i=m+1}^n s_i)| \leq 2\varepsilon\text{dist}(A,C)$, which will prove our dimensionality reduction result.

In order to do so, we can use the following 'weak triangle inequality', which is well known in the coreset literature, and can be generalized for other norms and distance functions (say, m-estimators).

LEMMA 7.1. *For any $1 > \varepsilon > 0$, a compact set $C \subseteq \mathbb{R}^n$, and $p, q \in \mathbb{R}^n$,*

$$|\text{dist}^2(p,C) - \text{dist}^2(q,C)| \leq \frac{12\|p-q\|^2}{\varepsilon} + \frac{\varepsilon}{2}\text{dist}^2(p,C).$$

8 Proof of Lemma 7.1

Proof. Using the triangle inequality,

(8.4)
$$\begin{aligned}
&|\text{dist}^2(p,B) - \text{dist}^2(q,B)| \\
&= |\text{dist}(p,B) - \text{dist}(q,B)| \cdot (\text{dist}(p,B) + \text{dist}(q,B)) \\
&\leq \|p-q\| \cdot (2\text{dist}(p,B) + \|p-q\|) \\
&\leq \|p-q\|^2 + 2\text{dist}(p,B)\|p-q\|.
\end{aligned}$$

Either $\text{dist}(p,B) \leq \|p-q\|/\varepsilon$ or $\|p-q\| < \varepsilon\text{dist}(p,B)$. Hence,

$$\text{dist}(p,B)\|p-q\| \leq \frac{\|p-q\|^2}{\varepsilon} + \varepsilon\text{dist}^2(p,B).$$

Combining the last inequality with (8.4) yields

$$\begin{aligned}
&|\text{dist}^2(p,B) - \text{dist}^2(q,B)| \\
&\leq \|p-q\|^2 + \frac{2\|p-q\|^2}{\varepsilon} + 2\varepsilon\text{dist}^2(p,B) \\
&\leq \frac{3\|p-q\|^2}{\varepsilon} + 2\varepsilon\text{dist}^2(p,B).
\end{aligned}$$

Finally, the lemma follows by replacing ε with $\varepsilon/4$. □

We can combine the above lemma with Corollary 7.1 by replacing ε in the corollary with $\varepsilon^2/30$ and summing the error of approximating an input point p by its projection q. If C is contained in a j^*-dimensional subspace, the error will be sufficiently small for $m \geq j^* + \lceil 30j^*/\varepsilon^2 \rceil - 1$. This is done in the proof of the theorem below.

THEOREM 8.1. *Let $A \in \mathbb{R}^{n\times n}$ be a matrix with singular value decomposition $A = UDV^T$ and let $\varepsilon > 0$. Let $j \geq 1$ be an integer, $j^* = j+1$ and $m = j^* + \lceil 30j^*/\varepsilon^2 \rceil - 1$ such that $m \leq n-1$. Furthermore, let $A' = UD^{(m)}V^T$, where $D^{(m)}$ is a diagonal matrix whose diagonal is the first m diagonal entries of D followed by $n-m$ zeros.*

Then for any compact set C, which is contained in a j-dimensional subspace, we have

$$\begin{aligned}
&\left|\text{dist}^2(A,C) - \left(\text{dist}^2(A',C) + \sum_{i=m+1}^n s_i\right)\right| \\
&\leq \varepsilon \cdot \text{dist}^2(A,C).
\end{aligned}$$

Proof. Suppose that $X \in \mathbb{R}^{n\times(j+1)}$ has orthonormal columns that span C. Let $Y \in \mathbb{R}^{n\times(n-(j+1))}$ denote a matrix whose orthonormal columns are also orthogonal to the columns of X. Since $\|AY\|^2$ is the sum of squared distances to the column space of X, we have

$$\|AY\|^2 \leq \text{dist}^2(A,C). \tag{8.5}$$

Fix an integer i, $1 \leq i \leq n$, and let p_i denote the ith row of the matrix AXX^T. That is, p_i is the projection of a row of A on X. Let p_i' denote the ith row of $A'XX^T$. Using Corollary 7.1, while replacing ε with $\varepsilon' = \varepsilon^2/30$

$$\begin{aligned}
&\sum_{i=1}^n \|p_i - p_i'\|^2 \\
&= \|UDVXX^T - UD^{(m)}VXX^T\|^2 \\
&\leq \frac{\varepsilon^2}{30} \cdot \|AY\|^2.
\end{aligned} \tag{8.6}$$

Notice that if u_i is the ith row of U, then the ith row of A can be written as $A_{i*} = u_i D V^T$, and thus for its projection p to X it holds $p = u_i D V^T X X^T$. Similarly, $p' = u_i D^{(m)} V^T X X^T$. Thus, the sum of the squared distances between all p and their corresponding p' is just $||UDV^T XX^T - UD^{(m)}V^T XX^T||_2^2$, which is bounded by $\varepsilon'||AY||^2$ by Corollary 7.1 and since we set $m = j^* + \lceil 30j^*/\varepsilon^2 \rceil - 1$ in the precondition of this Theorem. Together, we get

$$
\begin{aligned}
&\left|\mathrm{dist}^2(A,C) - \mathrm{dist}^2(A',C) - \sum_{i=j+1}^{n} s_i\right| \\
(8.7)\quad &= \Big|\|AY\|^2 - \|A'Y\|^2 \\
&\quad - \sum_{i=j+1}^{n} s_i + \sum_{p\in P}(\mathrm{dist}^2(p,C) - \mathrm{dist}^2(p',C))\Big| \\
(8.8)\quad &\leq \varepsilon'\|AY\|^2 \\
&\quad + \sum_{i=1}^{n}\left(\frac{12\|p_i - p_i'\|^2}{\varepsilon} + \frac{\varepsilon}{2}\cdot \mathrm{dist}^2(p_i,C)\right) \\
(8.9)\quad &\leq \left(\varepsilon' + \frac{12\varepsilon'}{\varepsilon} + \frac{\varepsilon}{2}\right)\mathrm{dist}^2(A,C) \\
&\leq \varepsilon \mathrm{dist}^2(A,C),
\end{aligned}
$$

where (8.7) follows from the Pythagorean Theorem, (8.8) follows from Lemma 7.1 and Corollary 6.2, and (8.9) follows from (8.5) and (8.6). □

9 Small Coresets for Projective Clustering

In this section we use the result of the previous section to prove that there is a strong coreset of size independent of the dimension of the space. In the last section, we showed that the projection A' of A is a coreset for A. A' still has n points, which are n-dimensional but lie in an m-dimensional subspace. To reduce the number of points, we want to apply known coreset constructions to A' within the low dimensional subspace. However, this would mean that our coreset only holds for centers that are also from the low dimensional subspace, but of course we want that the centers can be chosen from the full dimensional space. We get around this problem by applying the coreset constructions to a slightly larger space than the subspace that A' lives in. The following lemma provides us the necessary tool to complete the argumentation.

LEMMA 9.1. *Let L_1 be a d_1-dimensional subspace of $\mathbb{R}^n$ and let L be a (d_1+d_2)-dimensional subspace of $\mathbb{R}^n$ that contains L_1. Let L_2 be a d_2-dimensional subspace of $\mathbb{R}^n$. Then there is an orthonormal matrix U such that $Ux = x$ for any $x \in L_1$ and $Ux \in L$ for any $x \in L_2$.*

Proof. Let B_1 be an $n \times n$ matrix whose columns are an orthonormal basis of $\mathbb{R}^n$ such that the first d_1 columns span L_1 and the first $d_1 + d_2$ columns span L_2. Let B_2 an $n \times n$ matrix whose columns are an orthonormal basis of $\mathbb{R}^n$ such that the first d_1 columns are identical with B_1 and the first $d_1 + d_2$ columns span L_2. Define $U = B_1 B_2^T$. It is easy to verify that U satisfies the conditions of the lemma. □

Let L be any $m + j^*$-dimensional subspace that contains the m-dimensional subspace in which the points in A' lie, let C be an arbitrary compact set, which can, for example, be a set of centers (points, lines, subspaces, rings, etc.) and let L_2 be a j^*-dimensional subspace that contains C. If we now set $d_1 := m$ and $d_2 := j^*$ and let L_1 be the subspace spanned by the rows of A', then Lemma 9.1 implies that every given compact set (and so any set of centers) can be rotated into L. We could now proceed by computing a coreset in a $m + j^*$-dimensional subspace and then rotate every set of centers to this subspace. However, the last step is not necessary because of the following.

The mapping defined by any orthonormal matrix U is an isometry, and thus applying it does not change Euclidean distances. So, Lemma 9.1 implies that the sum of squared distances of C to the rows of A' is the same as the sum of squared distances of $U(C) := \{Ux : x \in C\}$ to A'. Now assume that we have a coreset for the subspace L. $U(C)$ is still a union of k subspaces, and it is in L. This implies that the sum of squared distances to $U(C)$ is approximated by the coreset. But this is identical to the sum of squared distances to C and so this is approximated by the coreset as well.

Thus, in order to construct a coreset for a set of n points in $\mathbb{R}^n$ we proceed as follows. In a first step we use the dimensionality reduction to reduce the input point set to a set of n points that lie in an m-dimensional subspace. Then we construct a coreset for a $(d + j^*)$-dimensional subspace that contains the low-dimensional point set. By the discussion above, this will be a coreset.

Finally, combining this with known results [21, 19, 36] we gain the following corolarry.

COROLLARY 9.1. *Let P be a set of points in $\mathbb{R}^d$, and $k, j \geq 0$ be a pair of integers. There is a set Q in $\mathbb{R}^d$, a weight function $w : Q \to [0, \infty)$ and a constant $c > 0$ such that the following holds.*

For every set B which is the union of k affine

j-subspaces of $\mathbb{R}^d$ we have

$$(1-\varepsilon)\sum_{p\in P}\text{dist}^2(p,B) \leq \sum_{p\in Q} w(p)\text{dist}^2(p,B)+c$$
$$\leq (1+\varepsilon)\sum_{p\in P}\text{dist}^2(p,B),$$

and

1. $|Q| = O(j/\varepsilon)$ *if* $k=1$
2. $|Q| = O(k^2/\varepsilon^4)$ *if* $j=0$
3. $|Q| = \text{poly}(2^k \log n, 1/\varepsilon)$ *if* $j=1$,
4. $|Q| = \text{poly}(2^{kj}, 1/\varepsilon)$ *if* $j,k>1$, *under the assumption that the coordinates of the points in P are integers between 1 and $n^{O(1)}$.*

In particular the size of Q is independent of d.

10 Fast, streaming, and parallel implementations

One major advantage of coresets is that they can be constructed in parallel as well as in a streaming setting.

That are can be constructed ('embarrassingly') in parallel is due to the fact that the union of coresets is again a coreset. More precisely, if a data set is split into ℓ subsets, and we compute a $(1+\varepsilon')$-coreset of size s for every subset, then the union of the ℓ coresets is a $(1+\varepsilon')$-coreset of the whole data set and has size $\ell \cdot s$. This is especially helpful if the data is already given in a distributed fashion on ℓ different machines. Then, every machine will simply compute a coreset. The small coresets can then be sent to a master device which approximately solves the optimization problem. Our coreset results for subspace approximation for one j-dimensional subspace, for k-means and for general projective clustering thus directly lead to distributed algorithms for solving these problems approximately.

If we want to compute a coreset with size independent of ℓ on the master device or in a parallel setting where the data was split and the coresets are later collected, then we can compute a $(1+\varepsilon')$-coreset of this merged coreset, gaining a $(1+\varepsilon')^2$-coreset. For $\varepsilon' := \varepsilon/3$, this results in a $(1+\varepsilon)$-coreset because $(1+\varepsilon/3)^2 < 1+\varepsilon$.

In the streaming setting, data points arrive one by one, in which it is impossible to remember the entire data set due to memory constraints. We state a failry general variant of the technique 'Merge & Reduce' [29] allowing us to use coresets in a stream while also being able to compute in parallel. For this, we define the notion of coreset schemes.

The following definition is a generalization of Definition 2, where the ground set $X = R^d \times [0,\infty) \times [0,\infty)$ is the set of points in $\mathbb{R}^d$, each assigned with multiplicative and additive weight, Q is all k-tuples of j-dimensional query subspaces in $\mathbb{R}^d$ for some fixed $j,k \geq 1$. For a given pair $j,k \geq 1$ of integers, a coreset scheme for the (j,k)-projective clustering problem $Alg(P,\varepsilon)$ is an algorithm whose input is a set $P \subseteq X$ of weighted points and an error parameter ε. It outputs a set C (called coreset) such that for every query subspace $L \in Q$, its cost

$$f(P,q) := \sum_{(p,w,w')\in P} w \cdot \text{dist}^2(p,L) + \sum_{(p,w,w')\in P} w'$$

approximated, up to a factor of $(1\pm\varepsilon)$, by $f(C,q)$.

In Definition 2, we denote $c = \sum_{(p,w,w')\in C} w'$ and assumed that P has no weights. However, in the merged-and-reduce approach that is described in this section, we apply the algorithm Alg recursively on its output coresets, and thus must assume that the input is also weighted. Since the distribution of the additive weights across the points has no effect on f, we can assume that only one arbitrary point in P has a positive additive weight.

Definition 3. (Coreset Scheme) *Let X and Q be two sets. Let $f : 2^X \times Q \to [0,\infty)$ be a function that maps every subset of X and $q \in Q$ to a positive real. An* ε-coreset scheme *$Alg(\cdot,\cdot)$ for (X,Q) is an algorithm that gets as input a set $P \subseteq X$ and a parameter $\varepsilon \in (0,1/10)$. It then returns a subset $C := A(P,\varepsilon)$ of X such that for every $q \in Q$ we have*

$$(1-\varepsilon)f(C,q) \leq f(P,q) \leq (1+\varepsilon)f(C,q)$$

The following theorem is a generalized and improved version of the coreset for streaming k-means from [29, Theorem 7.2]. The improvement is due to the use of $(\varepsilon/\log n)$-coresets rather than $(\varepsilon/\log^2 n)$-coresets, which are usually constructed in less time and space. Similar generalizations have been used in other papers. The proof uses certain composition properties that coresets satisfy.

Theorem 10.1. *Let X be a set representing a (possibly unbounded) stream of items, let Q be a set $\varepsilon \in (0,1/10)$ and let $Alg(\cdot,\cdot)$ be an ε-coreset scheme for (X,Q). Suppose that for every input set $P \subseteq X$ of $|P| \leq n'$ items, and every $\varepsilon \in (0,1/10)$ we can compute a coreset $Alg(P,\varepsilon)$ of size at most $a(\varepsilon,n')$ using at most $t(\varepsilon,n')$ time and $s(\varepsilon,n')$ space, and that there is an integer $m(\varepsilon)$ such that $a(\varepsilon,2m(\varepsilon)) \leq m(\varepsilon)$.*

Then, we can dynamically maintain an ε-coreset C for the n items seen so far at any point in the stream, where

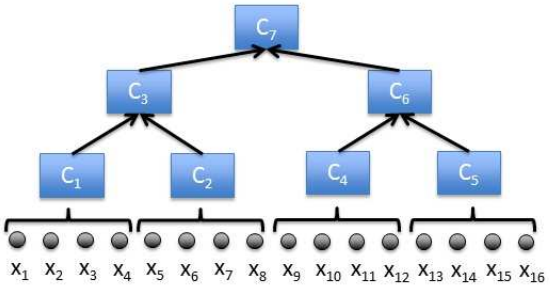

Figure 3: Tree construction for generating coresets in parallel or from a data stream. Black arrows indicate 'merge-and-reduce' operations. The (intermediate) coresets $C_1, \ldots, C_7$ are enumerated in the order in which they would be generated in the streaming case. In the parallel case, C_1, C_2, C_4 and C_5 would be constructed in parallel, followed by parallel construction of C_3 and C_6, finally resulting in C_7. The figure is taken from [33].

(i) it holds $|C| \leq a(\varepsilon'/9, m(\varepsilon') \cdot O(\log n))$ *for an* $\varepsilon' = O(\varepsilon/\log n)$.

(ii) The construction of C *takes at most*

$$s(\varepsilon', 2m(\varepsilon')) + s(\varepsilon/9, m(\varepsilon') \cdot O(\log n))$$

space with additional $m(\varepsilon') \cdot O(\log n)$ *items.*

(iii) Update time of C *per point insertion to the stream is*

$$t(\varepsilon', 2m(\varepsilon')) \cdot O(\log n) + t(\varepsilon/9, O(m(\varepsilon') \log n))$$

(iv) The amortized update time can be divided by M *using* $M \geq 1$ *processors in parallel.*

Proof. We first show how to construct an ε-coreset only for the first n items $p_1, \cdots, p_n$ in the stream, for a given $n \geq 2$, using space and update time as in *(i)* and *(ii)*.

Put $\varepsilon' = \varepsilon/(6\lceil \log n \rceil)$ and denote $m' = m(\varepsilon')$. The ε'-coreset of the first $2m' - 1$ items in the stream is simply their union. After the insertion of $p_{2m'}$ we replace the first $2m'$ items by their ε'-coreset C_1. The coreset for $p_1, \cdots, p_{2m'+i}$ is the union of C_1 and $p_{2m'+1}, \cdots, p_{2m'+i}$ for every $i = 1, \cdots, 2m$. When $p_{4m'}$ is inserted, we replace $p_{2m'+1}, \cdots, p_{4m'}$ by their ε'-coreset C_2. Using the assumption of the theorem, we have $|C_1| + |C_2| \leq 2m'$. We can thus compute an ε'-coreset of size at most m' *for the union of coresets* $C_1 \cup C_2$, and delete C_1 and C_2 from memory.

We continue to construct a binary tree of coresets as in Fig. 3. That is, every leaf and node contains at most m' items. Whenever we have two coresets in the same level, we replace them by a coreset in a higher level. The height of the tree is bounded from above by $\lceil \log n \rceil$. Hence, in every given moment during the streaming of n items, there are at most $O(\log n)$ ε'-coresets in memory, each of size at most m'. For any n, we can obtain a coreset for the first n points at the moment after they the nth point was added by computing an $(\varepsilon/3)$-coreset C for the union of these at most $O(\log n)$ coresets in the memory.

The approximation error of a coreset in the tree with respect to its leaves (the original items) is increased by a multiplicative factor of $(1+\varepsilon')$ in every tree level. Hence, the overall multiplicative error of the union of coresets in memory is $(1+\varepsilon')^{\lceil \log n \rceil}$. Using the definition of ε', we obtain

$$(1+\varepsilon')^{\lceil \log n \rceil} = (1 + \frac{\varepsilon}{6\lceil \log n \rceil})^{\lceil \log n \rceil} \leq e^{\frac{\varepsilon}{6}} < 1 + \varepsilon/3.$$

Since $(1 + \varepsilon/3)^2 \leq 1 + \varepsilon$ by the assumption $\varepsilon \in (0, 1/10)$, we have that C is an ε-coreset for the union of all the n items seen so far. We now prove claims (i)-(iv) for fixed n.

(i) C is an $\varepsilon/3$ coreset for $O(\log n)$ sets of size m', and thus $|C| = a(\varepsilon/3, m' \cdot O(\log n))$

(i) In every item insertion, there are $O(\log n)$ coresets in memory, each of size at most m', and additional $O(m)$ items at the leaves. Since we use additional space of $s(\varepsilon', 2m')$ for constructing a coreset in the tree, and $s(\varepsilon/3, m' \cdot O(\log n))$ space for constructing C, the overall space for the construction is bounded by

$$s(\varepsilon', 2m') + s(\varepsilon/3, m' \cdot O(\log n))$$

with additional $m' \cdot O(\log n)$ items.

(ii) When a new item is inserted, we need to apply the coreset construction at most $O(\log n)$ times, once for each level. Together with the computation time of C, the overall time is

$$t(\varepsilon', 2m') \cdot O(\log n) + t(\varepsilon/3, O(m' \log n))$$

per point insertion.

(iii) follows since the coresets can be computed in parallel for each level. More precisely, the $n/(2^i m')$ coresets in the ith level can be computed in parallel by dividing them into $\min\{M, n/(2^i m')\}$ machines.

The main problem with the above construction is that it assumes that n is known in advance. To remove this assumption, we compute the above coresets tree independently for batches (sets) with exponentially increasing size: a tree for the first batch of $n_1 = 2m'$ items in the stream (which consists of a single node), then a new tree for the next batch of

$n_2 = 4m'$ items, and in general a tree for ith batch of the following $n_i = 2^i m$ items in the stream for $i \geq 1$.

For every $i \geq 1$, let $\varepsilon_i = \varepsilon/(3c \log n_i)$ and compute an ε_i-coreset in each node of the ith tree, rather than ε'-coreset. After all the n_i items of the ith batch were read, its tree consists of a single (root) $(\varepsilon/3)$-coreset C_i of its n_i leaves, where $|C_i| = m(\varepsilon_i)$. After insertion of the nth item to the stream to the current jth batch, we have $j = O(\log n)$. We output an $(\varepsilon/3)$-coreset C for the union of previous coresets $C_1 \cup C_2 \cup \ldots \cup C_{j-1}$ with the $O(\log n)$ (ε_j)-coresets in the current (last) tree. Hence, C is an ε-coreset for the n items. Its size is union of the $O(\log n)$ $(\varepsilon/(3c \log n))$-coresets in its tree. Hence, C_j is an $(\varepsilon/3)$-coreset for the items that were read in the current batch, and thus $C_1 \cup C_2 \cup \cdots \cup C_j$ is an $(\varepsilon/3)$-coreset for all the n items. We then output an $(\varepsilon/3)$-coreset C for $C_1 \cup C_2 \cup \cdots \cup C_j$, which is an ε-coreset for the n items. The space and time bounds are dominated by the last batch of size $n_j \leq n$, so the time and space have the same bounds as above for a fixed set of size n.

For parallel construction of an unbounded stream, we simply send the ith point in the stream to machine number $(i \mod M)$ for every $i \geq 1$. Each machine computes a coreset to its given sub-stream, as explained above for a single machine. The coreset in every given moment is the union of current coresets in all the M machines. It may then be written in parallel to a shared memory on one of the machines.

□

11 Small Coresets for Other Dissimilarity Measures

In this section, we describe an alternative way to prove the existence of coresets independent of the number of input points and the dimension. It has an exponential dependency on ε^{-1} and thus leads to larger coresets for k-means. However, we show that the construction also works for a restricted class of μ-similar Bregman divergences, indicating that it is applicable to a wider class of distance functions.

Setting.

We are interested in formulating our strategy for more general distance measures than only l_2. Thus, for the formulation of the needed conditions, we just assume any distance function $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{\geq 0}$ which satisfies $d(0) = 0$. As usually, we use the abbreviations $\text{cost}(U) = \min_{z \in V} \sum_{x \in U} d(x, z)$, $\text{cost}(U, C) = \sum_{x \in U} \min_{c \in C} d(x, c)$ and $\text{opt}(U, k) = \min_{C \subset V, |C| = k} \text{cost}(U, C)$ for $U, C \subset \mathbb{R}^d$. We are interested in a coreset Z for P for approximate computation of $\text{opt}(P, C)$ given an arbitrary set C of k centers.

Clustering Features. A classical coreset is be a weighted set of points, for which the clustering cost is computed in a weighted fashion. In the sections above, we already saw that it can be beneficial to deviate from this definition. Here, we will construct a coreset consisting of so called *clustering features*. These are motivated by the well-known formula

$$\sum_{p \in P} d(p, z) = |P'| d(z, \mu(P)) + \sum_{p \in P} d(p, \mu(P)) \tag{11.10}$$

that holds for squared Euclidean distances as well as for Bregman divergences, if $\mu(P')$ denotes the centroid of P'. With help of 11.10, the sum of the distances of points in a set P' to a center c can be calculated exactly when only given the clustering features $|P|$, $\sum_{p \in P} p$ and $\sum_{p \in P} d(p, 0)$.

When given a CF and a set of centers, then the optimal choice to cluster the points represented by the clustering feature without splitting them is to assign them to the center closest to the centroid of the CF, which follows from 11.10. Thus, when clustering a set of CFs Z with a set of centers C we define the clustering cost $\text{cost}_{CF}(Z, C)$ as the cost for clustering every CF with the center closest to its centroid. We are interested in the following coreset type.

Definition 4. *Let P be a point set in $\mathbb{R}^d$. Let $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a distance measure that satisfies 11.10. A* CF-coreset *is a set of clustering features consisting of $|P|$, $\sum_{p \in P} p$ and $\sum_{p \in P} d(p, 0)$, such that*

$$|\text{cost}(P, C) - \text{cost}_{CF}(Z, C)| \leq \varepsilon \cdot \text{cost}(P, C)$$

holds for every set C of k centers.

In the following, we abbreviate CF-coreset by coreset. Notice that the error of the coreset compared to P is only induced by points which where summarized in one clustering feature but would optimally be clustered with different centers from C.

Construction and Sufficient Conditions.

Although we are mainly interested in CF-coresets, our algorithm will also work for other distance functions, also if they do not satisfy 11.10, if the distance function satisfies the following two conditions. This is independent of the actual definition of the coreset used, as long as the coreset approximates the cost as in Definition 4. We thus formulate the conditions and the algorithm for any distance function with any coreset notion, and cost_c denotes the cost function used for clustering with the unknown coreset type.

1. If $\text{cost}(P) \leq (1 + f_1(\varepsilon)) \sum_{i=1}^{k} \text{cost}(P_i)$ for all partitionings $P_1, \ldots, P_k$ of P into k subsets, then there exists a coreset Z of size $g(k, \varepsilon)$ such that for any set of k centers we have $|\text{cost}(P, C) - \text{cost}_c(Z, C)| \leq \varepsilon \, \text{cost}(P, C)$. So, if an optimal k-clustering of P is at most a $(1+\varepsilon)$-factor cheaper than the best 1-clustering, then this must induce a coreset for P.

2. If $\text{opt}(P', f_2(k)) \leq f_3(\varepsilon) \, \text{opt}(P, k)$ for $P' \subset P$, then there exists a set Z of size $h(f_2(k), \varepsilon)$ such that for any set of centers C we have $|\text{cost}(P', C) - \text{cost}_c(Z, C)| \leq \varepsilon \, \text{cost}(P, C)$.

Algorithm.

Given the above conditions, we use the following algorithm (the value of ν is defined later), P^* is the input point set:

Partition(P, k)

1. Let $C_1 \subset V$ be a set with $|C| = k$ that satisfies $\text{opt}(P, k) = \text{cost}(P, C_1)$. Consider the partitioning definined by$P_c = \{p \in P \mid n(c-p) = \min_{c' \in C} n(c' - p)\}$ into k subsets (ties broken arbitrarily).
   (a) If $\text{cost}(P) \leq (1+f_1(\varepsilon)) \sum_{c \in C} \text{cost}(P_c)$, stop.
   (b) Else, if this is the νth level of the recursion, still stop.
   If it is not, call Partition(P_c, k) for all $c \in C$ and then stop.

Note that the first time that we reach step 2, P is the input point set, and thus $\text{opt}(P, k) = \text{opt}(P^*, k)$ and $\sum_{c \in C} \text{cost}(P_c) \leq \text{opt}(P, k)/(1 + \varepsilon) = \text{opt}(P^*, k)/(1 + \varepsilon)$. Let $\mathcal{Q}$ denote the set of all subsets generated by the algorithm on level ν and for which the algorithm stopped in line 1b. On the ith level of the recursion, the sum of all sets is $\text{opt}(P^*, k)/(1 + f_1(\varepsilon))^i$. For $\nu = \lceil \log_{1+f_1(\varepsilon)} \frac{1}{f_3(\varepsilon)} \rceil$, this is smaller than $f_3(\varepsilon) \cdot \text{opt}(P^*, k)$. Thus, we have at most $f(k) := k^\nu$ sets in $\mathcal{Q}$, and $\sum_{U \in \mathcal{Q}} \text{cost}(U) \leq \varepsilon \cdot \text{opt}(P, k)$. By Condition 2, this implies the existence of a set Z of size $h(k^\nu, \varepsilon)$ which has an error of at most $\varepsilon \, \text{opt}(P^*, k)$.

For all sets where we stop in step 1a, Condition 1 directly gives a coreset of size $g(k, \varepsilon)$ for P. The union of these coresets give a coreset for the union of all sets which stoped in step 1a. Alltogether, they induce an error of less than $\varepsilon \, \text{opt}(P, k)$. Together with the $\varepsilon \, \text{opt}(P, k)$ error induced by the sets in $\mathcal{Q}$, this gives a total error of 2ε. So, if we start every thing with $\varepsilon/2$, we get a coreset for P with error $\varepsilon \, \text{opt}(P, k)$. The size of the coreset then is $k^\nu \cdot g(k, \varepsilon/2) + h(k^\nu, \varepsilon/2)$.

LEMMA 11.1. *If the cost function satisfies the above two conditions, then there exists a coreset of size* $k^\nu \cdot g(k, \varepsilon/2) + h(k^\nu, \varepsilon/2)$ *for* $\nu = \lceil \log_{1+f_1(\varepsilon/2)} \frac{1}{f_3(\varepsilon/2)} \rceil$.

For k-means, we can use clustering features and achieve that $g \equiv 1$ and $h(k^\nu, \varepsilon) = k^\nu$. Thus, the overall coreset size is $2k^{\log_{1+f_1(\varepsilon)} \frac{1}{f_3(\varepsilon)}}$. We do not present this in detail as the coreset is larger than the k-means coreset coming from our first construction. However, the proof can be deduced from the following proof for μ-similar bregman divergences, as the k-means case is easier.

11.1 Coresets for μ-similar Bregman divergences Let P be the input point set. Let $d_\phi : S \times S \to \mathbb{R}$ be a m-similar Bregman divergence, i. e., d_ϕ is defined on a convex set $S \subset \mathbb{R}^d$ and there exists a Mahalanobis distance d_B such that $m d_B(p, q) \leq d_\phi(p, q) \leq d_B(p, q)$ for all points $p, q \in \mathbb{R}^d$ and an $m \in (0, 1]$ (note that we use m-similar instead of μ-similar in order to prevent confusion with the centroid μ). We need the additional restriction on the convex set S that for every pair p, q of points from P, S contains all points within a ball of radius $(4/m\varepsilon) \cdot d(p, q)$ around p for a constant m, and we call such a set P-*covering*. Thus, in addition to the convex hull of the point set, a P-covering set may have to be larger by a factor dependent on m, ε and the diameter of P. Because of this additional restriction, our setting is much more restricted than in [6]. It is an interesting open question how to remove this restriction or even relax the μ-similarity further.

Notice that because d_B is a Mahalanobis distance there exists a regular matrix B with $d_B(x, y) = \|B(x - y)\|^2$ for all points $x, y \in \mathbb{R}^n$. In particular, $m \cdot \|B(x - y)\|^2 \leq d_\phi(x, y) \leq \|B(x - y)\|^2$. By [12], Bregman divergences (also if they are not m-similar) satisfy the Bregman version of equation (11.10), i. e.,

$$\sum_{p \in P} d_\phi(p, z) = \sum_{p \in P} d_\phi(p, \mu) + |P| d_\phi(\mu, z). \tag{11.11}$$

Condition 1. To show that Condition 1 holds, we set $f_1(\varepsilon) = \frac{1}{(1+\frac{4}{m \cdot \varepsilon})^2}$ and assume that we are given a point set S that satisfies that for every partitioning of S into k subsets $S_1, \ldots, S_k$ it holds that

$$\sum_{s \in S} d_\phi(s, \mu(S)) \leq (1 + f(\varepsilon)) \sum_{j=1}^{k} \sum_{x \in S_j} d_\phi(x, \mu(S_j))$$

$$\Leftrightarrow \sum_{j=1}^{k}\sum_{s\in S} d_\phi(s,\mu(S_j)) + \sum_{j=1}^{k}|S_j| d_\phi(\mu(S_j),\mu(S)) \leq (1+f(\varepsilon))\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j))$$

$$(4) \Leftrightarrow \sum_{j=1}^{k}|S_j| d_\phi(\mu(S_j),\mu(S)) \leq \frac{1}{(1+\frac{4}{m\cdot\varepsilon})^2}\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j)).$$

We show that this restricts the error of clustering all points in S with the same center, more specifically, with the center $c(\mu(S))$, the center closest to $\mu(S)$. To do so, we virtually add points to S. For every $j = 1,\dots,k$, we add one point with weight $\frac{1}{4}\varepsilon\cdot m\cdot|S_j|$ with coordinate $\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu(S)-\mu(S_j))$ to S_j. Notice that this points lies within the convex set A that d_B is defined on because we assumed that S is P-covering. The additional point shifts the centroid of S_j to $\mu(S)$ because

$$\frac{|S_j|\cdot\mu(S_j)+\frac{\varepsilon m}{4}|S_j|\left[\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu(S)-\mu(S_j))\right]}{(1+\frac{m\cdot\varepsilon}{4})|S_j|} = \frac{\frac{\varepsilon m}{4}|S_j|\left[\mu(S)+\frac{4}{m\cdot\varepsilon}\mu(S)\right]}{(1+\frac{m\cdot\varepsilon}{4})|S_j|} = \mu(S).$$

We name the set consisting of S_j together with the weighted added point S'_j and the union of all S'_j is S'. Now, clustering S' with center $c(\mu(S))$ is certainly an upper bound for the clustering cost of S with $c(\mu(S))$. Additionally, when clustering S'_j with only one center, then $c(\mu(S))$ is optimal, so clustering S'_j with $c(\mu(S_j))$ can only be more expensive. Thus, clustering all S'_j with the centers $c(\mu(S_j))$ gives an upper bound on the cost of clustering S with $c(\mu(S))$. So, to complete the proof, we have to upper bound the cost of clustering all S'_j with the respective centers $c(\mu(S_j))$. We do this by bounding the additional cost of clustering the added points with $c(\mu(S_j))$, which is

$$\sum_{j=1}^{k}\frac{\varepsilon m}{4}|S_j|\cdot d_\phi\left(\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu(S)-\mu(S_j)), c(\mu(S_j))\right)$$
$$\leq \sum_{j=1}^{k}\frac{\varepsilon m}{4}|S_j|\cdot\|B(\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu(S)-\mu(S_j))-c(\mu(S_j)))\|^2$$
$$= \|a\|^2$$

for the k-dimensional vector a defined by $a_j := \sqrt{\varepsilon m|S_j|/4}\|B(\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu(S)-\mu(S_j))-c(\mu(S_j)))\|$. By the triangle inequality, $a_j \leq \sqrt{\varepsilon m|S_j|/4}\|B((1+\frac{4}{m\varepsilon})(\mu(S)-\mu(S_j)))\| + \sqrt{\varepsilon m|S_j|/4}\|B(\mu(S_j)-c(\mu(S_j)))\| = b_j + d_j$ with $b_j = \sqrt{\varepsilon m|S_j|/4}\|B((1+\frac{4}{m\varepsilon})(\mu(S)-\mu(S_j)))\|$ and $d_j = \sqrt{\varepsilon m|S_j|/4}\|B(\mu(S_j)-c(\mu(S_j)))\|$. Then,

$$\|a\| \leq \|b+d\| \leq \|b\| + \|d\|,$$

where we use the triangle inequality again for the second inequality. Now we observe that

$$\|b\|^2 = \sum_{j=1}^{k}\frac{\varepsilon m}{4}|S_j|\|B((1+\frac{4}{\varepsilon m})(\mu(S)-\mu(S_j)))\|^2$$
$$= \frac{\varepsilon m}{4}\sum_{j=1}^{k}|S_j|(1+\frac{4}{m\varepsilon})^2\|B(\mu(S_j)-\mu(S))\|^2$$
$$\leq \frac{\varepsilon m}{4}(1+\frac{4}{m\varepsilon})^2\sum_{j=1}^{k}|S_j|\frac{1}{m}d_\phi(\mu(S_j),\mu(S))^2$$
$$\leq \frac{\varepsilon}{4}\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j)).$$

Additionally, by the definition of m-similarity and by Equation (11.11) it holds that

$$\|d\|^2 = \sum_{j=1}^{k}\frac{1}{4}\varepsilon m|S_j|\|B(\mu(S_j)-c(\mu(S_j)))\|^2$$
$$\leq \frac{\varepsilon}{4}\sum_{j=1}^{k}|S_j| d_\phi(\mu(S_j),c(\mu(S_j)))$$
$$\leq \frac{\varepsilon}{4}\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j)).$$

This implies that $\|a\| \leq \|b\| + \|d\| \leq 2\sqrt{\varepsilon}/2\sqrt{\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j))}$ and thus

$$\|a\|^2 \leq \varepsilon\sum_{j=1}^{k}\sum_{x\in S_j} d_\phi(x,\mu(S_j)).$$

This means that Condition 1 holds: If a k-clustering of S is not much cheaper than a 1-clustering, then assigning all points in S to the same center yields a $(1+\varepsilon)$-approximation for arbitrary center sets. Thus, we can use a clustering feature to store S, which means that $g(k, f'(\varepsilon^{-1})) \equiv 1$.

Condition 2. For the second condition, assume that $\mathcal{S}$ is a set of subsets of P representing the $f_2(k)$ subsets according to an optimal $f_2(k)$-clustering. Let a set C of k centers be given, and define the partitioning $S_1,\dots,S_k$ for every $S\in\mathcal{S}$ according to C as above.

By Equation (11.11) and by the precondition of Condition 2,

$$\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}|S_j|d_\phi(\mu(S_j),\mu(S))$$
$$=\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}\sum_{x\in S_j}d_\phi(x,\mu(S))-\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}\sum_{x\in S_j}d_\phi(x,\mu(S_j))$$
$$\leq f_3(\varepsilon)\cdot\mathrm{opt}(P,k).$$

We use the same technique as in the proof that Condition 1 holds. There are two changes: First, there are $|\mathcal{S}|$ sets where the centroids of the subsets must be moved to the centroid of the specific S (where in the above proof, we only had one set S). Second, the bound depends on $\mathrm{opt}(P,k)$ instead of $\sum_{S\in\mathcal{S}}$, so the approximation is dependent on $\mathrm{opt}(P,k)$ as well, but this is consistent with the statement in Condition 2. The complete proof that Condition 2 holds can be found in the appendix, it is very similar to the proof that Condition 1 holds with minor changes due to the two differences. We even set $f_3(\varepsilon)=f_1(\varepsilon)$.

THEOREM 11.1. *If $d_B : S\times S\to\mathbb{R}$ is a m-similar Bregman divergence on a convex and P-covering set S with $m\in(0,1]$, then there exists a coreset consisting of clustering features of constant size, i. e., the size only depends on k and ε.*

Proof. We have seen that the two conditions hold with $f_1(\varepsilon)=f_3(\varepsilon)=\frac{1}{(1+\frac{4}{m\cdot\varepsilon})^2}$, and $g\equiv 1$ and $h(k^\nu,\varepsilon)=k^\nu$. By Lemma 11.1, this implies a coreset size of

$$2k^\nu = 2k^{\lceil\log_{1+f_1(\varepsilon/2)}\frac{1}{f_3(\varepsilon/2)}\rceil}$$
$$= 2k^{\lceil\log_{1+\frac{1}{(1+\frac{8}{m\cdot\varepsilon})^2}}(1+\frac{8}{m\cdot\varepsilon})^2\rceil}$$
□

Acknowledgements. The project CG Learning acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 255827.

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 'Providing Information by Resource-Constrained Analysis', project A2.

References

[1] Community cleverness required. *Nature*, 455 (7209): 1. 4 September 2008. doi:10.1038/455001

[2] D. Seung and L. Lee. Algorithms for non-negative matrix factorization, *Advances in neural information processing systems*, volume 13, pp. 556–562, 2001.

[3] Blei, D.M. and Ng, A.Y. and Jordan, M.I. Latent Dirichlet Allocation *Journal of machine Learning research*, pp. 993–1022, 2003.

[4] IBM: What is big data?. Bringing big data to the enterprise. `ibm.com/software/data/bigdata/`, accessed on the 3rd of October 2012.

[5] Sandia sees data management challenges spiral. *HPC Projects*, 4 August 2009.

[6] M. Ackermann and J. Blömer. Coresets and approximate clustering for Bregman divergences. *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1088-1097, 2009.

[7] M. Ackermann, J. Blömer, and Christian Sohler. Clustering for metric and nonmetric distance measures. *ACM Transactions on Algorithms*, 6(4), 2010.

[8] Mahoney, M.W. Randomized Algorithms for Matrices and Data. *Foundations and Trends® in Machine Learning*, volume 3, II, pp.123–224, 2011, Now Publishers Inc.

[9] M. Ackermann, C. Lammersen, M. Märtens, C. Raupach, C. Sohler and K. Swierkot. StreamKM++: A Clustering Algorithms for Data Streams. *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*, pp. 173-187, 2010.

[10] P. Agarwal, S. Har-Peled and K. Varadarajan. Approximating extent measures of points. *Journal of the ACM*, 51(4): 606-635, 2004.

[11] Bentley, J.L. and Saxe, J.B. Decomposable searching problems I. Static-to-dynamic transformation, *Journal of Algorithms*, volume 1 (4), 301–358, 1980

[12] A. Banerjee, S. Merugu, I. S. Dhillon and J. Ghosh. Clustering with Bregman Divergences. *Journal of Machine Learning Research*, volume 6: 1705-1749, 2005.

[13] M. Bădoiu, S. Har-Peled, and P. Indyk. Approximate clustering via core-sets. *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 396–407, 2002.

[14] M. Beyer. Gartner Says Solving 'Big Data' Challenge Involves More Than Just Managing Volumes of Data. Gartner. Retrieved 13 July 2011. `http://www.gartner.com/it/page.jsp?id=1731916`.

[15] K. Chen. On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. *SIAM Journal on Computing*, 39(3): 923-947, 2009.

[16] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix Approximation and Projective Clustering via Volume Sampling. *Theory of Computing*, 2(1): 225-247, 2006.

[17] A. Deshpande, K. Varadarajan. Sampling-based dimension reduction for subspace approximation. *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 641-650, 2007.

[18] P. Drineas, A. Frieze, R. Kannan, S. Vempala,

V. Vinay. Clustering in large graphs via the Singular Value Decomposition. *Journal of Machine Learning*, volume 56 (1-3), pp. 9-33, 2004.

[19] D. Feldman, A. Fiat, M. Sharir. Coresets for-Weighted Facilities and Their Applications. *Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 315-324, 2006.

[20] D. Feldman, M. Monemizadeh, C. Sohler, and D. Woodruff. Coresets and Sketches for High Dimensional Subspace Approximation Problems. *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 630-649, 2010.

[21] D. Feldman, M. Langberg. A unified framework for approximating and clustering data. *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 569–578, 2011.

[22] D. Feldman, M. Monemizadeh and C. Sohler. A PTAS for k-means clustering based on weak coresets. Proceedings of the 23rd Annual ACM Symposium on Computational Geometry, pp. 11-18, 2007.

[23] D. Feldman, L. Schulman. Data reduction for weighted and outlier-resistant clustering. *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1343-1354, 2012.

[24] G. Frahling and C. Sohler. Coresets in dynamic geometric data streams. *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 209–217, 2005.

[25] G. Frahling, C. Sohler. A Fast k-Means Implementation Using Coresets. *International Journal of Computational Geometry and Applications*, 18(6): 605-625, 2008.

[26] S. Har-Peled. How to Get Close to the Median Shape. *CGTA*, 36 (1): 39-51, 2007.

[27] S. Har-Peled. No Coreset, No Cry. *Proceedings of the 24th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pp. 324-335, 2004.

[28] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Proceedings of the 21st Annual ACM Symposium on Computational Geometry*, pp. 126–134, 2005.

[29] S. Har-Peled and S. Mazumdar. Coresets for k-means and k-median clustering and their applications. *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 291–300, 2004.

[30] J. Hellerstein. Parallel Programming in the Age of Big Data. *Gigaom Blog*, 9th November, 2008.

[31] M. Hilbert and P. Lopez. The World's Technological Capacity to Store, Communicate, and Compute Information. *Science*, volume 332, No. 6025, pp. 60-65, 2011.

[32] A. Jacobs. The Pathologies of Big Data. ACMQueue, 6 July 2009.

[33] D. Feldman, A. Krause, and Matthew Faulkner. Scalable Training of Mixture Models via Coresets *Proc. 25th Conference on Neural Information Processing Systems (NIPS)* 2011

[34] J. Matoušek. On Approximate Geometric k-Clustering. Discrete & Computational Geometry, volume 24, No. 1, pp. 66-84, 2000.

[35] M. Langberg and L. J. Schulman. Universal epsilon-approximators for Integrals. *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 598-607, 2010.

[36] K. Varadarajan and X. Xiao. A near-linear algorithm for projective clustering integer points., *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2012

[37] O. J. Reichman, and M. B. Jones and M. P. Schildhauer. Challenges and Opportunities of Open Data in Ecology. *Science*, 331 (6018): 703-5. doi:10.1126/science.1197962, 2011.

[38] T. Segaran and J. Hammerbacher. Beautiful Data: The Stories Behind Elegant Data Solutions. O'Reilly Media. p. 257, 2009.

[39] N. Shyamalkumar, K. Varadarajan. Efficient subspace approximation algorithms. *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. pp. 532–540, 2007.

[40] T. White, Hadoop: The definitive guide. O'Reilly Media. 2012.

A Complete Proof for Condition 2.

For the second condition, assume that $\mathcal{S}$ is a set of subsets of P representing the $f_2(k)$ subsets according to an optimal $f_2(k)$-clustering. Let a set C of k centers be given, and define the partitioning $S_1, \ldots, S_k$ for every $S \in \mathcal{S}$ according to C as above. By Equation (11.11) and by the precondition of Condition 2,

$$\begin{aligned}
&\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}|S_j|d_\phi(\mu(S_j),\mu(S))\\
=&\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}\sum_{x\in S_j}d_\phi(x,\mu(S))-\sum_{S\in\mathcal{S}}\sum_{j=1}^{k}\sum_{x\in S_j}d_\phi(x,\mu(S_j))\\
\leq& f_3(\varepsilon)\cdot\mathrm{opt}(P,k).
\end{aligned}$$

We use the same technique as in the proof that Condition 1 holds. There are two changes: First, there are $|\mathcal{S}|$ sets where the centroids of the subsets must be moved to the centroid of the specific S (where in the above proof, we only had one set S). Second, the bound depends on $\mathrm{opt}(P,k)$ instead of $\sum_{S\in\mathcal{S}}$, so the approximation is dependent on $\mathrm{opt}(P,k)$ as well, but this is consistent with the statement in Condition 2. The complete proof that Condition 2 holds can be found in the appendix, it is very similar to the proof that Condition 1 holds with minor changes due to the two differences.

We set $f_3(\varepsilon) = f_1(\varepsilon)$ and again virtually add points. For each $S \in \mathcal{S}$ and each subset S_j of S, we add a point with weight $\frac{m\cdot\varepsilon}{4}|S_j|$ and coordinate $\mu(S)+\frac{4}{m\cdot\varepsilon}(\mu-\mu_j)$ to S_j. Notice that these points lie

within the convex set A that d_B is defined on because we assumed that S is P-covering.

We name the new sets S'_j, S' and $\mathcal{S}'$. Notice that the centroid of S'_j is now

$$\frac{|S_j| \cdot \mu(S_j) + \frac{\varepsilon m}{4}|S_j| \left[\mu(S) + \frac{4}{m \cdot \varepsilon} (\mu(S) - \mu(S_j))\right]}{(1 + \frac{m \cdot \varepsilon}{4})|S_j|}$$
$$= \mu(S)$$

in all cases. Again, clustering S' with $c(\mu(S))$ is an upper bound for the clustering cost of S with $c(\mu(S))$, and because the centroid of S'_j is $\mu(S)$, clustering every S'_j with $c(\mu(S_j))$ is an upper bound on clustering S with $c(\mu(S))$. Finally, we have to upper bound the cost of clustering all S'_j in all S with $c(\mu(S_j))$, which we again do by bounding the additional cost incurred by the added points. Adding this cost over all S yields

$$\begin{aligned}
&\sum_{S \in \mathcal{S}} \sum_{j=1}^{k} \frac{1}{4} \varepsilon m |S_j| \cdot d_\phi(\mu(S) \\
&\quad + \frac{4}{m \cdot \varepsilon} (\mu(S) - \mu(S_j)), c(\mu(S_j))) \\
\leq &\sum_{S \in \mathcal{S}} \sum_{j=1}^{k} \frac{\varepsilon m}{4} |S_j| \cdot \|B(\mu(S) \\
&\quad + \frac{4}{m \cdot \varepsilon} (\mu(S) - \mu(S_j)) - c(\mu(S_j)))\|^2 = \|a\|^2.
\end{aligned}$$

For the last equality, we define $|\mathcal{S}|$ vectors a^S by $a_j^S := \sqrt{\varepsilon m |S_j|/4} \|B(\mu(S) + \frac{4}{m \cdot \varepsilon} (\mu(S) - \mu(S_j)) - c(\mu(S_j)))\|$ and concatenate them in arbitrary but fixed order to get a $k \cdot |\mathcal{S}|$ dimensional vector a. By the triangle inequality, $a_j^S \leq \sqrt{\varepsilon m |S_j|/4} \|B((1 + \frac{4}{m\varepsilon})(\mu(S) - \mu(S_j)))\| + \sqrt{\varepsilon m |S_j|/4} \|B(\mu(S_j) - c(\mu(S_j)))\| = b_j^S + d_j^S$ with $b_j^S = \sqrt{\varepsilon m |S_j|/4} \|B((1 + \frac{4}{m\varepsilon})(\mu(S) - \mu(S_j)))\|$ and $d_j^S = \sqrt{\varepsilon m |S_j|/4} \|B(\mu(S_j) - c(\mu(S_j)))\|$. Define b and d by concatenating the vectors b^S and d^S, respectively, in the same order as used for a. Then we can again conclude that

$$\|a\| \leq \|b + d\| \leq \|b\| + \|d\|,$$

where we use the triangle inequality for the second inequality. Now we observe that

$$\begin{aligned}
\|b\|^2 = &\sum_{S \in \mathcal{S}} \sum_{j=1}^{k} \frac{\varepsilon m}{4} |S_j| \|B((1 + \frac{4}{\varepsilon m})(\mu(S) - \mu(S_j)))\|^2 \\
= &\frac{\varepsilon m}{4} \sum_{S \in \mathcal{S}} \sum_{j=1}^{k} |S_j| (1 + \frac{4}{m\varepsilon})^2 \|B(\mu(S_j) - \mu(S))\|^2 \\
\leq &\frac{\varepsilon m}{4} (1 + \frac{4}{m\varepsilon})^2 \sum_{S \in \mathcal{S}} \sum_{j=1}^{k} |S_j| \frac{1}{m} d_\phi(\mu(S_j), \mu(S))^2 \\
\leq &\frac{\varepsilon}{4} \operatorname{opt}(P, k).
\end{aligned}$$

Additionally, by the definition of m-similarity and by Equation (11.11) it holds that

$$\begin{aligned}
\|d\|^2 = &\sum_{S \in \mathcal{S}} \sum_{j=1}^{k} \frac{1}{4} \varepsilon m |S_j| \|B(\mu(S_j) - c(\mu(S_j)))\|^2 \\
\leq &\frac{\varepsilon}{4} \sum_{S \in \mathcal{S}} \sum_{j=1}^{k} |S_j| d_\phi(\mu(S_j), c(\mu(S_j))) \\
\leq &\frac{\varepsilon}{4} \sum_{S \in \mathcal{S}} \sum_{j=1}^{k} \sum_{x \in S_j} d_\phi(x, \mu(S_j)).
\end{aligned}$$

This implies that $\|a\| \leq \|b\| + \|d\| \leq 2\sqrt{\varepsilon}/2\sqrt{\operatorname{opt}(P, k)}$ and thus

$$\|a\|^2 \leq \varepsilon \operatorname{opt}(P, k).$$