Random Projections for k-means Clustering

Christos Boutsidis
Department of Computer Science
RPI

Anastasios Zouzias
Department of Computer Science
University of Toronto

Petros Drineas
Department of Computer Science
RPI

Abstract

This paper discusses the topic of dimensionality reduction for k-means clustering. We prove that any set of n points in d dimensions (rows in a matrix $A \in \mathbb{R}^{n \times d}$) can be projected into $t = \Omega(k/\varepsilon^2)$ dimensions, for any $\varepsilon \in (0, 1/3)$, in $O(nd\lceil\varepsilon^{-2}k/\log(d)\rceil)$ time, such that with constant probability the optimal k-partition of the point set is preserved within a factor of $2+\varepsilon$. The projection is done by post-multiplying A with a $d \times t$ random matrix R having entries $+1/\sqrt{t}$ or $-1/\sqrt{t}$ with equal probability. A numerical implementation of our technique and experiments on a large face images dataset verify the speed and the accuracy of our theoretical results.

1 Introduction

The k-means clustering algorithm [16] was recently recognized as one of the top ten data mining tools of the last fifty years [20]. In parallel, random projections (RP) or the so-called Johnson-Lindenstrauss type embeddings [12] became popular and found applications in both theoretical computer science [2] and data analytics [4]. This paper focuses on the application of the random projection method (see Section 2.3) to the k-means clustering problem (see Definition 1). Formally, assuming as input a set of n points in d dimensions, our goal is to randomly project the points into $\tilde{d}$ dimensions, with $\tilde{d} \ll d$, and then apply a k-means clustering algorithm (see Definition 2) on the projected points. Of course, one should be able to compute the projection fast without distorting significantly the "clusters" of the original point set. Our algorithm (see Algorithm 1) satisfies both conditions by computing the embedding in time linear in the size of the input and by distorting the "clusters" of the dataset by a factor of at most $2+\varepsilon$, for some $\varepsilon \in (0, 1/3)$ (see Theorem 1). We believe that the high dimensionality of modern data will render our algorithm useful and attractive in many practical applications [9].

Dimensionality reduction encompasses the union of two different approaches: *feature selection*, which embeds the points into a low-dimensional space by selecting actual dimensions of the data, and *feature extraction*, which finds an embedding by constructing new artificial features that are, for example, linear combinations of the original features. Let A be an $n \times d$ matrix containing n d-dimensional points ($A_{(i)}$ denotes the i-th point of the set), and let k be the number of clusters (see also Section 2.2 for more notation). We slightly abuse notation by also denoting by A the n-point set formed by the rows of A. We say that an embedding $f : A \to \mathbb{R}^{\tilde{d}}$ with $f(A_{(i)}) = \tilde{A}_{(i)}$ for all $i \in [n]$ and some $\tilde{d} < d$, preserves the clustering structure of A within a factor ϕ, for some $\phi \geq 1$, if finding an optimal clustering in $\tilde{A}$ and plugging it back to A is only a factor of ϕ worse than finding the optimal clustering directly in A. Clustering optimality and approximability are formally presented in Definitions 1 and 2, respectively. Prior efforts on designing provably accurate dimensionality reduction methods for k-means clustering include: *(i)* the Singular Value Decomposition (SVD), where one finds an embedding with image $\tilde{A} = U_k\Sigma_k \in \mathbb{R}^{n \times k}$ such that the clustering structure is preserved within a factor of two; *(ii)* random projections, where one projects the input points into $t = \Omega(\log(n)/\varepsilon^2)$ dimensions such that with constant probability the clustering structure is preserved within a factor of $1+\varepsilon$ (see Section 2.3); *(iii)* SVD-based feature selection, where one can use the SVD to find $c = \Omega(k\log(k/\varepsilon)/\varepsilon^2)$ actual features, i.e. an embedding with image $\tilde{A} \in \mathbb{R}^{n \times c}$ containing (rescaled) columns from A, such that with constant probability the clustering structure is preserved within a factor of $2+\varepsilon$. These results are summarized in Table 1. A head-to-head comparison of our algorithm with existing results allows us to claim the following improvements: *(i)*

Year	Ref.	Description	Dimensions	Time	Accuracy
1999	[6]	SVD - feature extraction	k	$O(nd\min\{n,d\})$	2
-	Folklore	RP - feature extraction	$\Omega(\log(n)/\varepsilon^2)$	$O(nd\lceil\varepsilon^{-2}\log(n)/\log(d)\rceil)$	$1+\varepsilon$
2009	[5]	SVD - feature selection	$\Omega(k\log(k/\varepsilon)/\varepsilon^2)$	$O(nd\min\{n,d\})$	$2+\varepsilon$
2010	This paper	RP - feature extraction	$\Omega(k/\varepsilon^2)$	$O(nd\lceil\varepsilon^{-2}k/\log(d)\rceil)$	$2+\varepsilon$

Table 1: Dimension reduction methods for k-means. In the RP methods the construction is done with random sign matrices and the mailman algorithm (see Sections 2.3 and 3.1, respectively).

reduce the running time by a factor of $\min\{n,d\}\lceil\varepsilon^2\log(d)/k\rceil$, while losing only a factor of ε in the approximation accuracy and a factor of $1/\varepsilon^2$ in the dimension of the embedding; *(ii)* reduce the dimension of the embedding and the running time by a factor of $\log(n)/k$ while losing a factor of one in the approximation accuracy; *(iii)* reduce the dimension of the embedding by a factor of $\log(k/\varepsilon)$ and the running time by a factor of $\min\{n,d\}\lceil\varepsilon^2\log(d)/k\rceil$, respectively. Finally, we should point out that other techniques, for example the Laplacian scores [10] or the Fisher scores [7], are very popular in applications (see also surveys on the topic [8, 13]). However, they lack a theoretical worst case analysis of the form we describe in this work.

2 Preliminaries

We start by formally defining the k-means clustering problem using matrix notation. Later in this section, we precisely describe the approximability framework adopted in the k-means clustering literature and fix the notation.

Definition 1. [THE K-MEANS CLUSTERING PROBLEM]
Given a set of n points in d dimensions (rows in an $n\times d$ matrix A) and a positive integer k denoting the number of clusters, find the $n\times k$ indicator matrix X_{opt} such that

$$X_{opt} = \arg\min_{X\in\mathcal{X}} \left\|A - XX^\top A\right\|_\mathrm{F}^2. \tag{1}$$

Here $\mathcal{X}$ denotes the set of all $n\times k$ indicator matrices X. The functional $F(A,X) = \left\|A - XX^\top A\right\|_\mathrm{F}^2$ is the so-called k-means objective function. An $n\times k$ indicator matrix has exactly one non-zero element per row, which denotes cluster membership. Equivalently, for all $i=1,\ldots,n$ and $j=1,\ldots,k$, the i-th point belongs to the j-th cluster if and only if $X_{ij} = 1/\sqrt{z_j}$, where z_j denotes the number of points in the corresponding cluster. Note that $X^\top X = I_k$, where I_k is the $k\times k$ identity matrix.

2.1 Approximation Algorithms for k-means clustering

Finding X_{opt} is an NP-hard problem even for $k=2$ [3], thus research has focused on developing approximation algorithms for k-means clustering. The following definition captures the framework of such efforts.

Definition 2. [K-MEANS APPROXIMATION ALGORITHM]
An algorithm is a "γ-approximation" for the k-means clustering problem ($\gamma\geq 1$) if it takes inputs A and k, and returns an indicator matrix X_γ that satisfies with probability at least $1-\delta_\gamma$,

$$\left\|A - X_\gamma X_\gamma^\top A\right\|_\mathrm{F}^2 \leq \gamma \min_{X\in\mathcal{X}} \left\|A - XX^\top A\right\|_\mathrm{F}^2. \tag{2}$$

In the above, $\delta_\gamma\in[0,1)$ is the failure probability of the γ-approximation k-means algorithm.

For our discussion, we fix the γ-approximation algorithm to be the one presented in [14], which guarantees $\gamma = 1+\varepsilon'$ for any $\varepsilon'\in(0,1]$ with running time $O(2^{(k/\varepsilon')^{O(1)}}dn)$.

2.2 Notation

Given an $n\times d$ matrix A and an integer k with $k<\min\{n,d\}$, let $U_k\in\mathbb{R}^{n\times k}$ (resp. $V_k\in\mathbb{R}^{d\times k}$) be the matrix of the top k left (resp. right) singular vectors of A, and let $\Sigma_k\in\mathbb{R}^{k\times k}$ be a diagonal matrix containing the top

k singular values of A in non-increasing order. If we let ρ be the rank of A, then $A_{\rho-k}$ is equal to $A - A_k$, with $A_k = U_k \Sigma_k V_k^\top$. By $A_{(i)}$ we denote the i-th row of A. For an index i taking values in the set $\{1, \ldots, n\}$ we write $i \in [n]$. We denote, in non-increasing order, the non-negative singular values of A by $\sigma_i(A)$ with $i \in [\rho]$. $\|A\|_\mathrm{F}$ and $\|A\|_2$ denote the Frobenius and the spectral norm of a matrix A, respectively. $A^\dagger$ denotes the pseudo-inverse of A, i.e. the unique $d \times n$ matrix satisfying $A = AA^\dagger A$, $A^\dagger A A^\dagger = A^\dagger$, $(AA^\dagger)^\top = AA^\dagger$, and $(A^\dagger A)^\top = A^\dagger A$. Note also that $\left\|A^\dagger\right\|_2 = \sigma_1(A^\dagger) = 1/\sigma_\rho(A)$ and $\|A\|_2 = \sigma_1(A) = 1/\sigma_\rho(A^\dagger)$. A useful property of matrix norms is that for any two matrices C and T of appropriate dimensions, $\|CT\|_\mathrm{F} \leq \|C\|_\mathrm{F} \|T\|_2$; this is a stronger version of the standard submultiplicavity property. We call P a projector matrix if it is square and $P^2 = P$. We use $\mathbb{E}[Y]$ and $\mathrm{Var}[Y]$ to take the expectation and the variance of a random variable Y and $\mathbb{P}(e)$ to take the probability of an event e. We abbreviate "independent identically distributed" to "i.i.d." and "with probability" to "w.p.". Finally, all logarithms are base two.

2.3 Random Projections

A classical result of Johnson and Lindenstrauss states that any n-point set in d dimensions - rows in a matrix $A \in \mathbb{R}^{n \times d}$ - can be linearly projected into $t = \Omega(\log(n)/\varepsilon^2)$ dimensions while preserving pairwise distances within a factor of $1 \pm \varepsilon$ using a random orthonormal matrix [12]. Subsequent research simplified the proof of the above result by showing that such a projection can be generated using a $d \times t$ random Gaussian matrix R, i.e., a matrix whose entries are i.i.d. Gaussian random variables with zero mean and variance $1/\sqrt{t}$ [11]. More precisely, the following inequality holds with high probability over the randomness of R,

$$(1-\varepsilon)\left\|A_{(i)} - A_{(j)}\right\|_2 \leq \left\|A_{(i)}R - A_{(j)}R\right\|_2 \leq (1+\varepsilon)\left\|A_{(i)} - A_{(j)}\right\|_2. \tag{3}$$

Notice that such an embedding $\tilde{A} = AR$ preserves the metric structure of the point-set, so it also preserves, within a factor of $1+\varepsilon$, the optimal value of the k-means objective function of A. Achlioptas proved that even a (rescaled) random sign matrix suffices in order to get the same guarantees as above [1], an approach that we adopt here (see step two in Algorithm 1). Moreover, in this paper we will heavily exploit the structure of such a random matrix, and obtain, as an added bonus, savings on the computation of the projection.

3 A random-projection-type k-means algorithm

Algorithm 1 takes as inputs the matrix $A \in \mathbb{R}^{n \times d}$, the number of clusters k, an error parameter $\varepsilon \in (0, 1/3)$, and some γ-approximation k-means algorithm. It returns an indicator matrix $X_{\tilde{\gamma}}$ determining a k-partition of the rows of A.

Input: $n \times d$ matrix A (n points, d features), number of clusters k, error parameter $\varepsilon \in (0, 1/3)$, and γ-approximation k-means algorithm.
Output: Indicator matrix $X_{\tilde{\gamma}}$ determining a k-partition on the rows of A.

1. Set $t = \Omega(k/\varepsilon^2)$, i.e. set $t = t_o \geq ck/\varepsilon^2$ for a sufficiently large constant c.
2. Compute a random $d \times t$ matrix R as follows. For all $i \in [d]$, $j \in [t]$
$$R_{ij} = \begin{cases} +1/\sqrt{t}, \text{w.p. } 1/2, \\ -1/\sqrt{t}, \text{w.p. } 1/2. \end{cases}$$
3. Compute the product $\tilde{A} = AR$.
4. Run the γ-approximation algorithm on $\tilde{A}$ to obtain $X_{\tilde{\gamma}}$; Return the indicator matrix $X_{\tilde{\gamma}}$

Algorithm 1: A random projection algorithm for k-means clustering.

3.1 Running time analysis

Algorithm 1 reduces the dimensions of A by post-multiplying it with a random sign matrix R. Interestingly, any "random projection matrix" R that respects the properties of Lemma 2 with $t = \Omega(k/\varepsilon^2)$ can be used in this step. If R is constructed as in Algorithm 1, one can employ the so-called mailman algorithm for matrix multiplication [15] and compute the product AR in $O(nd\lceil \varepsilon^{-2}k/\log(d)\rceil)$ time. Indeed, the mailman algorithm computes (after preprocessing [1]) a matrix-vector product of any d-dimensional vector (row of A) with an $d \times \log(d)$ sign matrix in $O(d)$ time. By partitioning the columns of our $d \times t$ matrix R into $\lceil t/\log(d)\rceil$ blocks, the claim follows. Notice that when $k = O(\log(d))$, then we get an - almost - linear time complexity $O(nd/\varepsilon^2)$. The latter assumption is reasonable in our setting since the need for dimension reduction in k-means clustering arises usually in high-dimensional data (large d). Other choices of R would give the same approximation results; the time complexity to compute the embedding would be different though. A matrix where each entry is a random Gaussian variable with zero mean and variance $1/\sqrt{t}$ would imply an $O(knd/\varepsilon^2)$ time complexity (naive multiplication). In our experiments in Section 5 we experiment with the matrix R described in Algorithm 1 and employ MatLab's matrix-matrix BLAS implementation to proceed in the third step of the algorithm. We also experimented with a novel MatLab/C implementation of the mailman algorithm but, in the general case, we were not able to outperform MatLab's built-in routines (see section 5.2).

Finally, note that any γ-approximation algorithm may be used in the last step of Algorithm 1. Using, for example, the algorithm of [14] with $\gamma = 1+\varepsilon$ would result in an algorithm that preserves the clustering within a factor of $2+\varepsilon$, for any $\varepsilon \in (0, 1/3)$, running in time $O(nd\lceil \varepsilon^{-2}k/\log(d)\rceil + 2^{(k/\varepsilon)^{O(1)}}kn/\varepsilon^2)$. In practice though, the Lloyd algorithm [16, 17] is very popular and although it does not admit a worst case theoretical analysis, it empirically does well. We thus employ the Lloyd algorithm for our experimental evaluation of our algorithm in Section 5. Note that, after using the proposed dimensionality reduction method, the cost of the Lloyd heuristic is only $O(nk^2/\varepsilon^2)$ per iteration. This should be compared to the cost of $O(knd)$ per iteration if applied on the original high dimensional data.

4 Main Theorem

Theorem 1 is our main quality-of-approximation result for Algorithm 1. Notice that if $\gamma = 1$, i.e. if the k-means problem with inputs $\tilde{A}$ and k is solved exactly, Algorithm 1 guarantees a distortion of at most $2+\varepsilon$, as advertised.

Theorem 1. *Let the $n \times d$ matrix A and the positive integer $k < \min\{n, d\}$ be the inputs of the k-means clustering problem. Let $\varepsilon \in (0, 1/3)$ and assume access to a γ-approximation k-means algorithm. Run Algorithm 1 with inputs A, k, ε, and the γ-approximation algorithm in order to construct an indicator matrix $X_{\tilde{\gamma}}$. Then with probability at least $0.97 - \delta_\gamma$,*

$$\left\| A - X_{\tilde{\gamma}} X_{\tilde{\gamma}}^\top A \right\|_{\mathrm{F}}^2 \leq (1 + (1+\varepsilon)\gamma) \left\| A - X_{opt} X_{opt}^\top A \right\|_{\mathrm{F}}^2 . \tag{4}$$

Proof of Theorem 1

The proof of Theorem 1 employs several results from [19] including Lemma 6, 8 and Corollary 11. We summarize these results in Lemma 2 below. Before employing Corollary 11, Lemma 6, and Lemma 8 from [19] we need to make sure that the matrix R constructed in Algorithm 1 is consistent with Definition 1 and Lemma 5 in [19]. Theorem 1.1 of [1] immediately shows that the random sign matrix R of Algorithm 1 satisfies Definition 1 and Lemma 5 in [19].

Lemma 2. *Assume that the matrix R is constructed by using Algorithm 1 with inputs A, k and ε.*

1. *Singular Values Preservation: For all $i \in [k]$ and w.p. at least 0.99,*
$$|1 - \sigma_i(V_k^\top R)| \leq \varepsilon.$$
2. *Matrix Multiplication: For any two matrices $S \in \mathbb{R}^{n\times d}$ and $T \in \mathbb{R}^{d\times k}$,*
$$\mathbb{E}\left[\left\| ST - SRR^\top T \right\|_{\mathrm{F}}^2\right] \leq \frac{2}{t} \|S\|_{\mathrm{F}}^2 \|T\|_{\mathrm{F}}^2 .$$
3. *Moments: For any $C \in \mathbb{R}^{n\times d}$: $\mathbb{E}\left[\|CR\|_{\mathrm{F}}^2\right] = \|C\|_{\mathrm{F}}^2$ and Var $[\|CR\|_{\mathrm{F}}] \leq 2\|C\|_{\mathrm{F}}^4/t$.*

The first statement above assumes c being sufficiently large (see step 1 of Algorithm 1). We continue with several novel results of general interest.

[1]Reading the input $d \times \log d$ sign matrix requires $O(d \log d)$ time. However, in our case we only consider multiplication with a *random* sign matrix, therefore we can avoid the preprocessing step by directly computing a random *correspondence* matrix as discussed in [15, Preprocessing Section].

Lemma 3. *Under the same assumptions as in Lemma 2 and w.p. at least* 0.99,

$$\left\|(V_k^\top R)^\dagger - (V_k^\top R)^\top\right\|_2 \;\le\; 3\varepsilon. \tag{5}$$

Proof. Let $\Phi = V_k^\top R$; note that Φ is a $k \times t$ matrix and the SVD of Φ is $\Phi = U_\Phi \Sigma_\Phi V_\Phi^\top$, where U_Φ and Σ_Φ are $k \times k$ matrices, and V_Φ is a $t \times k$ matrix. By taking the SVD of $(V_k^\top R)^\dagger$ and $(V_k^\top R)^\top$ we get

$$\left\|(V_k^\top R)^\dagger - (V_k^\top R)^\top\right\|_2 \;=\; \left\|V_\Phi \Sigma_\Phi^{-1} U_\Phi^\top - V_\Phi \Sigma_\Phi U_\Phi^\top\right\|_2 \;=\; \left\|V_\Phi (\Sigma_\Phi^{-1} - \Sigma_\Phi) U_\Phi^\top\right\|_2 \;=\; \left\|\Sigma_\Phi^{-1} - \Sigma_\Phi\right\|_2,$$

since V_Φ and $U_\Phi^\top$ can be dropped without changing any unitarily invariant norm. Let $\Psi = \Sigma_\Phi^{-1} - \Sigma_\Phi$; Ψ is a $k \times k$ diagonal matrix. Assuming that, for all $i \in [k]$, $\sigma_i(\Phi)$ and $\tau_i(\Psi)$ denote the i-th largest singular value of Φ and the i-th diagonal element of Ψ, respectively, it is

$$\tau_i(\Psi) \;=\; \frac{1 - \sigma_i(\Phi)\sigma_{k+1-i}(\Phi)}{\sigma_{k+1-i}}.$$

Since Ψ is a diagonal matrix,

$$\|\Psi\|_2 \;=\; \max_{1 \le i \le k} \tau_i(\Psi) \;=\; \max_{1 \le i \le k} \frac{1 - \sigma_i(\Phi)\sigma_{k+1-i}(\Phi)}{\sigma_{k+1-i}(\Phi)}.$$

The first statement of Lemma 2, our choice of $\varepsilon \in (0, 1/3)$ and elementary calculations suffice to conclude the proof. ☐

Lemma 4. *Under the same assumptions as in Lemma 2 and for any* $n \times d$ *matrix* C *w.p. at least* 0.99,

$$\|CR\|_\mathrm{F} \;\le\; \sqrt{(1+\varepsilon)}\,\|C\|_\mathrm{F}. \tag{6}$$

Proof. Notice that there exists a sufficiently large constant c such that $t \ge ck/\varepsilon^2$. Then, setting $Z = \|CR\|_\mathrm{F}^2$, using the third statement of Lemma 2, the fact that $k \ge 1$, and Chebyshev's inequality we get

$$\mathbb{P}\left(|Z - \mathbb{E}[Z]| \ge \varepsilon \|C\|_\mathrm{F}^2\right) \;\le\; \frac{\mathrm{Var}[Z]}{\varepsilon^2 \|C\|_\mathrm{F}^4} \;\le\; \frac{2\|C\|_\mathrm{F}^4}{t\varepsilon^2 \|C\|_\mathrm{F}^4} \;\le\; \frac{2}{ck} \le\; 0.01.$$

The last inequality follows assuming c sufficiently large. Finally, taking square root on both sides concludes the proof. ☐

Lemma 5. *Under the same assumptions as in Lemma 2 and w.p. at least* 0.97,

$$A_k = (AR)(V_k^\top R)^\dagger V_k^\top + E, \tag{7}$$

where E *is an* $n \times d$ *matrix with* $\|E\|_\mathrm{F} \le 4\varepsilon \|A - A_k\|_\mathrm{F}$.

Proof. Since $(AR)(V_k^\top R)^\dagger V_k^\top$ is an $n \times d$ matrix, let us write $E = A_k - (AR)(V_k^\top R)^\dagger V_k^\top$. Then, setting $A = A_k + A_{\rho-k}$, and using the triangle inequality we get

$$\|E\|_\mathrm{F} \;\le\; \left\|A_k - A_k R (V_k^\top R)^\dagger V_k^\top\right\|_\mathrm{F} + \left\|A_{\rho-k} R (V_k^\top R)^\dagger V_k^\top\right\|_\mathrm{F}.$$

The first statement of Lemma 2 implies that $\mathrm{rank}(V_k^\top R) = k$ thus $(V_k^\top R)(V_k^\top R)^\dagger = I_k$, where I_k is the $k \times k$ identity matrix. Replacing $A_k = U_k \Sigma_k V_k^\top$ and setting $(V_k^\top R)(V_k^\top R)^\dagger = I_k$ we get that

$$\left\|A_k - A_k R (V_k^\top R)^\dagger V_k^\top\right\|_\mathrm{F} \;=\; \left\|A_k - U_k \Sigma_k V_k^\top R (V_k^\top R)^\dagger V_k^\top\right\|_\mathrm{F} \;=\; \left\|A_k - U_k \Sigma_k V_k^\top\right\|_\mathrm{F} \;=\; 0.$$

To bound the second term above, we drop $V_k^\top$, add and subtract the matrix $A_{\rho-k} R (V_k^\top R)^\top V_k^\top$, and use the triangle inequality and submultiplicativity:

$$\begin{aligned}
\left\|A_{\rho-k} R (V_k^\top R)^\dagger V_k^\top\right\|_\mathrm{F} &\le \left\|A_{\rho-k} R (V_k^\top R)^\top\right\|_\mathrm{F} + \left\|A_{\rho-k} R \left((V_k^\top R)^\dagger - (V_k^\top R)^\top\right)\right\|_\mathrm{F} \\
&\le \left\|A_{\rho-k} R R^\top V_k\right\|_\mathrm{F} + \|A_{\rho-k} R\|_\mathrm{F} \left\|(V_k^\top R)^\dagger - (V_k^\top R)^\top\right\|_2.
\end{aligned}$$

Now we will bound each term individually. A crucial observation for bounding the first term is that $A_{\rho-k}V_k = U_{\rho-k}\Sigma_{\rho-k}V_{\rho-k}^\top V_k = \mathbf{0}$ by orthogonality of the columns of V_k and $V_{\rho-k}$. This term now can be bounded using the second statement of Lemma 2 with $S = A_{\rho-k}$ and $T = V_k$. This statement, assuming c sufficiently large, and an application of Markov's inequality on the random variable $\left\|A_{\rho-k}RR^\top V_k - A_{\rho-k}V_k\right\|_\mathrm{F}$ give that w.p. at least 0.99,

$$\left\|A_{\rho-k}RR^\top V_k\right\|_\mathrm{F} \;\leq\; 0.5\varepsilon \left\|A_{\rho-k}\right\|_\mathrm{F}. \tag{8}$$

The second two terms can be bounded using Lemma 3 and Lemma 4 on $C = A_{\rho-k}$. Hence by applying a union bound on Lemma 3, Lemma 4 and Inq. (8), we get that w.p. at least 0.97,

$$\begin{aligned}
\|E\|_\mathrm{F} &\leq \left\|A_{\rho-k}RR^\top V_k\right\|_\mathrm{F} + \left\|A_{\rho-k}R\right\|_\mathrm{F}\left\|\left(V_k^\top R\right)^\dagger - \left(V_k^\top R\right)^\top\right\|_2 \\
&\leq 0.5\varepsilon\left\|A_{\rho-k}\right\|_\mathrm{F} + \sqrt{(1+\varepsilon)}\left\|A_{\rho-k}\right\|_\mathrm{F}\cdot 3\varepsilon \\
&\leq 0.5\varepsilon\left\|A_{\rho-k}\right\|_\mathrm{F} + 3.5\varepsilon\left\|A_{\rho-k}\right\|_\mathrm{F} \\
&= 4\varepsilon\cdot\left\|A_{\rho-k}\right\|_\mathrm{F}.
\end{aligned}$$

The last inequality holds thanks to our choice of $\varepsilon \in (0, 1/3)$. □

Proposition 6. *A well-known property connects the SVD of a matrix and k-means clustering. Recall Definition 1, and notice that $X_{opt}X_{opt}^\top A$ is a matrix of rank at most k. From the SVD optimality we immediately get that*

$$\left\|A_{\rho-k}\right\|_\mathrm{F}^2 \;=\; \left\|A - A_k\right\|_\mathrm{F}^2 \;\leq\; \left\|A - X_{opt}X_{opt}^\top A\right\|_\mathrm{F}^2. \tag{9}$$

4.1 The proof of Eqn. (4) of Theorem 1

We start by manipulating the term $\left\|A - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top A\right\|_\mathrm{F}^2$ in Eqn. (4). Replacing A by $A_k + A_{\rho-k}$, and using the Pythagorean theorem (the subspaces spanned by the components $A_k - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top A_k$ and $A_{\rho-k} - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top A_{\rho-k}$ are perpendicular) we get

$$\left\|A - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top A\right\|_\mathrm{F}^2 \;=\; \underbrace{\left\|(I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top)A_k\right\|_\mathrm{F}^2}_{\theta_1^2} + \underbrace{\left\|(I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top)A_{\rho-k}\right\|_\mathrm{F}^2}_{\theta_2^2}. \tag{10}$$

We first bound the second term of Eqn. (10). Since $I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top$ is a projector matrix, it can be dropped without increasing a unitarily invariant norm. Now Proposition 6 implies that

$$\theta_2^2 \;\leq\; \left\|A_{\rho-k}\right\|_\mathrm{F}^2 \;\leq\; \left\|A - X_{opt}X_{opt}^\top A\right\|_\mathrm{F}^2. \tag{11}$$

We now bound the first term of Eqn. (10):

$$\begin{aligned}
\theta_1 &\leq \left\|(I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top)AR(V_kR)^\dagger V_k^\top\right\|_\mathrm{F} \;+\; \|E\|_\mathrm{F} && (12)\\
&\leq \left\|(I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top)AR\right\|_\mathrm{F}\left\|(V_kR)^\dagger\right\|_2 \;+\; \|E\|_\mathrm{F} && (13)\\
&\leq \sqrt{\gamma}\left\|(I - X_{opt}X_{opt}^\top)AR\right\|_\mathrm{F}\left\|(V_kR)^\dagger\right\|_2 \;+\; \|E\|_\mathrm{F} && (14)\\
&\leq \sqrt{\gamma}\sqrt{(1+\varepsilon)}\left\|(I - X_{opt}X_{opt}^\top)A\right\|_\mathrm{F}\frac{1}{1-\varepsilon} \;+\; 4\varepsilon\left\|(I - X_{opt}X_{opt}^\top)A\right\|_\mathrm{F} && (15)\\
&\leq \sqrt{\gamma}(1+2.5\varepsilon)\left\|(I - X_{opt}X_{opt}^\top)A\right\|_\mathrm{F} + \sqrt{\gamma}\,4\varepsilon\left\|(I - X_{opt}X_{opt}^\top)A\right\|_\mathrm{F} && (16)\\
&\leq \sqrt{\gamma}(1+6.5\varepsilon)\left\|(I - X_{opt}X_{opt}^\top)A\right\|_\mathrm{F} && (17)
\end{aligned}$$

In Eqn. (12) we used Lemma 5, the triangle inequality, and the fact that $I - \tilde{X}_\gamma\tilde{X}_\gamma^\top$ is a projector matrix and can be dropped without increasing a unitarily invariant norm. In Eqn. (13) we used submultiplicativity (see Section 2.2) and the fact that $V_k^\top$ can be dropped without changing the spectral norm. In Eqn. (14) we replaced $X_{\tilde{\gamma}}$ by X_{opt} and the factor $\sqrt{\gamma}$ appeared in the first term. To better understand this step, notice that $X_{\tilde{\gamma}}$ gives a γ-approximation to the optimal k-means clustering of the matrix AR, and any other $n \times k$ indicator matrix (for example, the matrix X_{opt}) satisfies

$$\left\|\left(I - X_{\tilde{\gamma}}X_{\tilde{\gamma}}^\top\right)AR\right\|_\mathrm{F}^2 \leq \gamma \min_{X\in\mathcal{X}}\left\|(I - XX^\top)AR\right\|_\mathrm{F}^2 \leq \gamma\left\|\left(I - X_{opt}X_{opt}^\top\right)AR\right\|_\mathrm{F}^2.$$

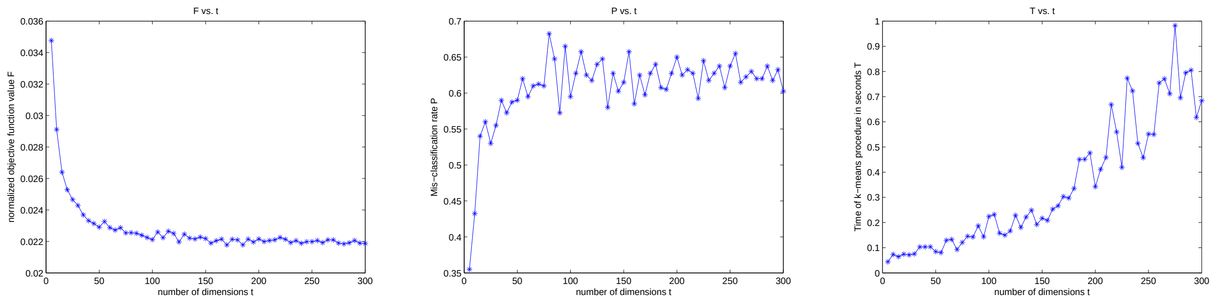

Figure 1: The results of our experiments after running Algorithm 1 with $k = 40$ on the face images collection.

In Eqn. (15) we used Lemma 4 with $C = (I - X_{opt}X_{opt}^\top)A$, Lemma 3 and Proposition 6. In Eqn. (16) we used the fact that $\gamma \geq 1$ and that for any $\varepsilon \in (0, 1/3)$ it is $(\sqrt{1+\varepsilon})/(1-\varepsilon) \leq 1 + 2.5\varepsilon$. Taking squares in Eqn. (17) we get

$$\theta_1^2 \;\leq\; \gamma(1 + 28\varepsilon)\left\|(I - X_{opt}X_{opt}^\top)A\right\|_{\mathrm{F}}^2.$$

Finally, rescaling ε accordingly and applying the union bound on Lemma 5 and Definition 2 concludes the proof.

5 Experiments

This section describes an empirical evaluation of Algorithm 1 on a face images collection. We implemented our algorithm in MatLab and compared it against other prominent dimensionality reduction techniques such as the Local Linear Embedding (LLE) algorithm and the Laplacian scores for feature selection. We ran all the experiments on a Mac machine with a dual core 2.26 Ghz processor and 4 GB of RAM. Our empirical findings are very promising indicating that our algorithm and implementation could be very useful in real applications involving clustering of large-scale data.

5.1 An application of Algorithm 1 on a face images collection

We experiment with a face images collection. We downloaded the images corresponding to the ORL database from [21]. This collection contains 400 face images of dimensions 64×64 corresponding to 40 different people. These images form 40 groups each one containing exactly 10 different images of the same person. After vectorizing each 2-D image and putting it as a row vector in an appropriate matrix, one can construct a 400×4096 image-by-pixel matrix A. In this matrix, objects are the face images of the ORL collection while features are the pixel values of the images. To apply the Lloyd's heuristic on A, we employ MatLab's function $kmeans$ with the parameter determining the maximum number of repetitions setting to 30. We also chose a deterministic initialization of the Lloyd's iterative E-M procedure, i.e. whenever we call $kmeans$ with inputs a matrix $\tilde{A} \in R^{400 \times \tilde{d}}$, with $\tilde{d} \geq 1$, and the integer $k = 40$, we initialize the cluster centers with the 1-st, 11-th,..., 391-th rows of $\tilde{A}$, respectively. Note that this initialization corresponds to picking images from the forty different groups of the available collection, since the images of every group are stored sequentially in A. We evaluate the clustering outcome from two different perspectives. First, we measure and report the objective function F of the k-means clustering problem. In particular, we report a normalized version of F, i.e. $\tilde{F} = F/||A||_F^2$. Second, we report the mis-classification accuracy of the clustering result. We denote this number by P ($0 \leq P \leq 1$), where $P = 0.9$, for example, implies that 90% of the objects were assigned to the correct cluster after the application of the clustering algorithm. In the sequel, we first perform experiments by running Algorithm 1 with everything fixed but t, which denotes the dimensionality of the projected data. Then, for four representative values of t, we compare Algorithm 1 with three other dimensionality reduction methods as well with the approach of running the Lloyd's heuristic on the original high dimensional data.

We run Algorithm 1 with $t = 5, 10, ..., 300$ and $k = 40$ on the matrix A described above. Figure 1 depicts the results of our experiments. A few interesting observations are immediate. First, the normalized objective function $\tilde{F}$ is a piece-wise non-increasing function of the number of dimensions t. The decrease in $\tilde{F}$ is large in the first few choices

	t = 10		**t = 20**		**t = 50**		**t = 100**	
	P	F	P	F	P	F	P	F
SVD	0.5900	0.0262	0.6750	0.0268	0.7650	0.0269	0.6500	0.0324
LLE	0.6500	0.0245	0.7125	0.0247	0.7725	0.0258	0.6150	0.0337
LS	0.3400	0.0380	0.3875	0.0362	0.4575	0.0319	0.4850	0.0278
HD	0.6255	0.0220	0.6255	0.0220	0.6255	0.0220	0.6255	0.0220
RP	0.4225	0.0283	0.4800	0.0255	0.6425	0.0234	0.6575	0.0219

Table 2: Numerics from our experiments with five different methods.

of t; then, increasing the number of dimensions t of the projected data decreases $\tilde{F}$ by a smaller value. The increase of t seems to become irrelevant after around $t = 90$ dimensions. Second, the mis-classification rate P is a piece-wise non-decreasing function of t. The increase of t seems to become irrelevant again after around $t = 90$ dimensions. Another interesting observation of these two plots is that the mis-classification rate is not directly relevant to the objective function F. Notice, for example, that the two have different behavior from $t = 20$ to $t = 25$ dimensions. Finally, we report the running time T of the algorithm which includes only the clustering step. Notice that the increase in the running time is - almost - linear with the increase of t. The non-linearities in the plot are due to the fact that the number of iterations that are necessary to guarantee convergence of the Lloyd's method are different for different values of t. This observation indicates that small values of t result to significant computational savings, especially when n is large. Compare, for example, the one second running time that is needed to solve the k-means problem when $t = 275$ against the 10 seconds that are necessary to solve the problem on the high dimensional data. To our benefit, in this case, the multiplication AR takes only 0.1 seconds resulting to a total running time of 1.1 seconds which corresponds to an almost 90% speedup of the overall procedure.

We now compare our algorithm against other dimensionality reduction techniques. In particular, in this paragraph we present head-to-head comparisons for the following five methods: (i) SVD: the Singular Value Decomposition (or Principal Components Analysis) dimensionality reduction approach - we use MatLab's $svds$ function; (ii) LLE: the famous Local Linear Embedding algorithm of [18] - we use the MatLab code from [23] with the parameter K determining the number of neighbors setting equal to 40; (iii) LS: the Laplacian score feature selection method of [10] - we use the MatLab code from [22] with the default parameters[2]; (v) HD: we run the k-means algorithm on the High Dimensional data; and (vi) RP: the random projection method we proposed in this work - we use our own MatLab implementation. The results of our experiments on A, $k = 40$ and $t = 10, 20, 50, 100$ are shown in Table 2. In terms of computational complexity, for example $t = 50$, the time (in seconds) needed for all five methods (only the dimension reduction step) are $T_{SVD} = 5.9$, $T_{LLE} = 4.4$, $T_{LS} = 0.32$, $T_{HD} = 0$, and $T_{RP} = 0.03$. Notice that our algorithm is much faster than the other approaches while achieving worse ($t = 10, 20$), slightly worse ($t = 50$) or slightly better ($t = 100$) approximation accuracy results.

5.2 A note on the mailman algorithm for matrix-matrix and matrix-vector multiplication

In this section, we compare three different implementations of the third step of Algorithm 1. As we already discussed in Section 3.1, the mailman algorithm is asymptotically faster than naively multiplying the two matrices A and R. In this section we want to understand whether this asymptotic behavior of the mailman algorithm is indeed achieved in a practical implementation. We compare three different approaches for the implementation of the third step of our algorithm: the first is MatLab's function $times(A, R)$ (MM1); the second exploits the fact that we do not need to explicitly store the whole matrix R, and that the computation can be performed on the fly (column-by-column) (MM2); the last is the mailman algorithm [15] (see Section 3.1 for more details). We implemented the last two algorithms in C using MatLab's MEX technology. We observed that when A is a vector $(n = 1)$, then the mailman algorithm is indeed faster than (MM1) and (MM2) as it is also observed in the numerical experiments of [15]. Moreover, it's worth-noting that (MM2) is also superior compared to (MM1). On the other hand, our best implementation of the mailman algorithm for matrix-matrix operations is inferior to both (MM1) and (MM2) for any $10 \leq n \leq 10,000$. Based on these findings, we chose to use (MM1) for our experimental evaluations.

Acknowledgments: Christos Boutsidis was supported by NSF CCF 0916415 and a Gerondelis Foundation Fellowship; Petros Drineas was partially supported by an NSF CAREER Award and NSF CCF 0916415.

[2]In particular, we run $W = constructW(A)$; $Scores = LaplacianScore(A, W)$;

References

[1] D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Science*, 66(4):671–687, 2003.

[2] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *ACM Symposium on Theory of Computing (STOC)*, pages 557–563, 2006.

[3] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–248, 2009.

[4] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pages 245–250, 2001.

[5] C. Boutsidis, M. W. Mahoney, and P. Drineas. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

[6] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 291–299, 1999.

[7] D. Foley and J. Sammon. An optimal set of discriminant vectors. *IEEE Transactions on Computers*, C-24(3):281–289, March 1975.

[8] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[9] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge. In *Advances in Neural Information Processing Systems (NIPS)*, pages 545–552. 2005.

[10] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. In *Advances in Neural Information Processing Systems (NIPS) 18*, pages 507–514. 2006.

[11] P. Indyk and R. Motwani Approximate nearest neighbors: towards removing the curse of dimensionality. In *ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[12] W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189-206):1–1, 1984.

[13] E. Kokiopoulou, J. Chen and Y. Saad. Trace optimization and eigenproblems in dimension reduction methods. Numerical Linear Algebra with Applications, to appear.

[14] A. Kumar, Y. Sabharwal, and S. Sen. A simple linear time $(1+\varepsilon)$-approximation algorithm for k-means clustering in any dimensions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 454–462, 2004.

[15] E. Liberty and S. Zucker. The Mailman algorithm: A note on matrix-vector multiplication. *Information Processing Letters*, 109(3):179–182, 2009.

[16] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[17] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 165–176, 2006.

[18] S. Roweis, and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. Science, 290:5500, pages 2323-2326, 2000.

[19] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 329–337, 2006.

[20] X. Wu et al. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[21] `http://www.cs.uiuc.edu/~dengcai2/Data/FaceData.html`

[22] `http://www.cs.uiuc.edu/~dengcai2/Data/data.html`

[23] `http://www.cs.nyu.edu/~roweis/lle/`