Iterative random projections for high-dimensional data clustering

Ângelo Cardoso, Andreas Wichert

INESC-ID Lisboa and Instituto Superior Técnico, Technical University of Lisbon
Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal

Abstract

In this text we propose a method which efficiently performs clustering of high-dimensional data. The method builds on random projection and the K-means algorithm. The idea is to apply K-means several times, increasing the dimensionality of the data after each convergence of K-means. We compare the proposed algorithm on four high-dimensional datasets, image, text and two synthetic, with K-means clustering using a single random projection and K-means clustering of the original high-dimensional data. Regarding time we observe that the algorithm reduces drastically the time when compared to K-means on the original high-dimensional data. Regarding mean squared error the proposed method reaches a better solution than clustering using a single random projection. More notably in the experiments performed it also reaches a better solution than clustering on the original high-dimensional data.

Keywords: clustering, K-means, high-dimensional data, random projections

1. Introduction

The K-means algorithm [10], given a set of n points in R^d and an integer K, finds the K centers, such that the total squared error between each the of the n points and its closest center is minimized. The algorithm starts by

Email addresses: angelo.cardoso@ist.utl.pt (Ângelo Cardoso), andreas.wichert@ist.utl.pt (Andreas Wichert)

initializing the centers randomly. Each cluster center is defined by the empirical mean of its points. At each iteration, each of the points is assigned to its closest center and afterwards the centers are recalculated. Each K-means iteration reduces the total squared error. When the algorithm converges it reaches a minimum, however there is no guarantee that it is global.

Random projection [7] is used for projecting high-dimensional data into low-dimensional subspaces while approximately preserving the Euclidean distance between the points. Its application in computer science is wide, including nearest neighbor search, clustering and classification. In this paper we propose a novel method for clustering using random projection which gradually increases the dimensionality of the data in successive applications of K-means. We call the method iterative random projections K-means (IRP K-means). We compare it to related approaches which use a single random projection and to K-means clustering in the original high-dimensional space. Experiments on an image dataset and a text dataset indicate that the proposed algorithm improves significantly the mean squared error (MSE) in the original space when compared to a single random projection. The empirical results also show that by gradually increasing the dimensionality of the data we can reach a solution with a lower MSE than clustering on the original high-dimensional space. IRP K-Means is related to simulated annealing clustering [12] which avoids local minimums. However such approach relatively to K-means greatly increases the running time [2].

2. Random projection for K-means clustering

Random projection works by projecting the high dimensional data into a lower dimensional random subspace. We randomly create h vectors which define random projection matrix R. This idea is based on the Johnson-Lindenstrauss lemma [7] whose intuition is: if n points in vector space of dimension d are projected onto a randomly selected subspace of suitably high dimensions h, then the Euclidean distance between the points are approximately preserved. In [5] more tight bounds are given, more precisely the theorem is the following:

Theorem 2.1. *For any $0 < \epsilon < 1$ and any integer n, let h be a positive integer such that*

$$h \geq 4 \left(\frac{\varepsilon^2}{2} - \frac{\varepsilon^3}{3} \right)^{-1} \ln n.$$

Then for any set V of n points in R^d, there is a map $f : R^d \to R^h$ such that for all u, $v \in V$,

$$(1 - \varepsilon)\|u - v\|_2 \leq \|f(u) - f(v)\|_2 \leq (1 + \varepsilon)\|u - v\|_2.$$

Furthermore, this map can be found in randomized polynomial time.

Given a random matrix $P_{d\times h}$ and a data matrix $X_{n\times d}$, then the projection of X into P, is given by

$$X^{RP}_{n\times h} = X_{n\times d}P_{d\times h}. \tag{1}$$

Several alternatives have been proposed for generating the random matrix [1, 9] which reduce the computational cost of projecting the data, namely, using integers and sparseness in matrix P. The generated random matrix P is usually not orthogonal. Making R orthogonal is computationally expensive. However, in a high-dimensional space, there exists a much larger number of almost orthogonal than orthogonal directions [6], therefore vectors with random directions are close enough to be orthogonal.

2.1. A K-means based algorithm using random projection

In this section we describe an algorithm for K-means clustering using random projection which is related to several previous works [4, 11, 3]. By clustering the data on lower dimensional space, the computational cost of each K-means iteration can be greatly reduced, while still finding a solution which is related to clustering on the original high-dimensional space.

The algorithm starts by initializing the cluster membership G randomly. The cluster membership is set by selecting randomly K points as the clusters centers. Then we generate a random matrix P to project the data. We project the data $X_{n\times d}$ to D dimensions ($D < d$) using a random projection defined by matrix $P_{d\times D}$. Using the projected data $X^{RP}_{n\times D}$ and G. We define the initial cluster centers C^{RP} as the mean of each cluster in X^{RP}, which at initialization is simply one point per cluster. Finally we run K-means until convergence or some stopping criterion is met on X_{RP} using C_{RP} as initialization, obtaining K clusters defined by the cluster membership G. The detailed description is given in Algorithm 1.

Algorithm 1: Random Projection K-means

input : dimension D
data $X_{n\times d}$
number of clusters K
output: cluster membership G
begin
Set the cluster membership G as K randomly selected points from X.
Set a random matrix $P_{d\times D}$.
Set $X^{RP}_{n\times D} = XP$.
Set $C^{RP}_{k\times D}$ by calculating the mean of each cluster in X^{RP} according to G.
Obtain G with K-means on X^{RP} with C^{RP} as initialization.
return G

2.2. A K-means based algorithm using iterative dimension random projection

The intuition behind the algorithm is that by iteratively increasing the dimension of the space, we can construct a solution with increasingly more detail, while avoiding local minimums in the original space. The gradual construction of the solution saves iterations in later dimensions, that are more costly. This is analogous to cooling in simulated annealing clustering [12]. The lower the dimension, the greater the probability of assigning a point to a wrong cluster, i.e. a cluster whose center is not the closest according to the Euclidean distance in the original space. Lower dimensions are equivalent to a higher temperature in simulated annealing clustering. However unlike simulated annealing K-means, iterations in higher temperatures are faster because they are computed in lower dimensions instead of the original dimension.

The algorithm is related to random projection K-means (see Algorithm 1) but instead of using a projection for a single dimension, we project the data and cluster it several epochs, increasing the dimension of the projection in each of the epochs. The clusters obtained in a given dimension are used to initialize the clusters in the following dimension.

The algorithm starts in dimension D_1, by projecting the high-dimensional data X into a space of dimension $D_1 (D_1 < d)$ using a random projection defined by P_1, obtaining X^{RP_1}. In the first dimension D_1 we initialize the centroids randomly by selecting K points as centers. By applying K-means

in X^{RP_1} we obtain the new cluster membership G, which will then be used to initialize K-means in the next dimension (D_2). Using the cluster membership G obtained from K-means in dimension D_1 and X^{RP_2}, we recalculate the centroids now in dimension $D_2(D_1 \leq D_2 < d$, to obtain the new initial centroids C^{RP_2} now in a new space of dimension D_2. Now in D_2, we apply K-means again using C^{RP_2} as initialization. Then analogously repeat the same process until reaching the last dimension $D_l(D_1 \leq D_2 \leq D_l < d)$ returning the cluster membership from D_l. The relation $D_1 \leq D_2 \leq ... \leq D_l < d$ is heuristic and analogous to simulated annealing cooling, $D_1, D_2, ..., D_l$ can therefore take any values. The detailed description is given in Algorithm 2.

Algorithm 2: Iterative random projections K-means

input : list of dimensions $D_{a=1,...,l}$
data $X_{n \times d}$
number of clusters K
output: cluster membership G
begin
  Set the cluster membership G as K randomly selected points from X.
  for $a = 1$ *to* l **do**
    Set a random matrix $P_a(d \times D_a)$.
    Set $X^{RP_a}(n \times D_a) = XP_a$.
    Set $C^{RP_a}(k \times D_a)$ by calculating the mean of each cluster in X^{RP_a} according to G.
    Obtain G with K-means on X^{RP_a} with C^{RP_a} as initialization.
  return G

3. Experiments

We evaluate the performance of the proposed method [1] on four high-dimensional datasets, an image, a text and two synthetic datasets.

We measure the clustering performance by the MSE, i.e. the K-means objective function. Regarding time performance we report the running time and the number of K-means iterations. Each entry in random matrix P is sampled i.i.d from the standard Gaussian $\mathcal{N}(0, 1)$, which gives the same bound for h as Theorem 2.1 as discussed in [5, 1].

[1] An implementation can be obtained from the authors

We use the same implementation of K-means for all the reported methods. All the experiments were run in Matlab on a 64 bit machine with a 2.1 GHz dual-core CPU and 4 GB of RAM.

3.1. Datasets

We use the AT&T Database of Faces[2] (formerly ORL Database of Faces), which contains 400 images from 40 subjects. The size of each image is 92×112 pixels with 256 gray levels, therefore 10304 dimensions. We use $K = 40$ in the experiments since that is the number of subjects.

For the text experiments we use a dataset containing papers from NIPS 1-12[3]. The text dataset contains 2484 documents with 14036 words, therefore 14036 dimensions. The original data contains the term count for each document. We normalize all document vectors to unit length, by dividing the term count by the total number of terms for a given document. The dataset is sparse, on average for each document only $3,93\%$ of the terms are present. For this dataset we do not know the correct K value, we use $K = 10$ in the experiments.

We also generated two synthetic datasets both with 10000 dimensions. For the first dataset each cluster j is sampled from a multivariate Gaussian distribution $\mathcal{N}(\mu_j, \sigma^2)$ being μ_j sampled from a multivariate Gaussian distribution $\mathcal{N}(0, 1)$ and σ^2 is a diagonal matrix ($d \times$d) (common to all clusters) where each entry is sampled from a multivariate Uniform distribution $\mathcal{U}(0, 2)$. For the second dataset each cluster j is sampled from a multivariate Uniform distribution $\mathcal{U}(c_j - d_j, c_j + d_j)$ where c_j is sampled from a multivariate Uniform distribution $\mathcal{U}(-1, 1)$ and d_j is sampled from a multivariate Uniform distribution $\mathcal{U}(0, 2)$. Each dataset contains 1000 points randomly sampled from each cluster with probability $1/K$. We use $K = 20$ for both synthetic datasets. The MSE of assigning all points to the correct cluster is 6.61×10^3 for the Gaussian dataset and 4.39×10^3 for the Uniform dataset.

In the experiments we will compare K-means on the original high-dimensional data (referred by *Classic*), Algorithm 1 (referred by *RP*) and Algorithm 2 (referred by *IRP*). All experiments for all algorithms report the MSE on the original high-dimensional space. To evaluate if the differences between the algorithms are statistically significant we use a two sample t-test with

[2] AT&T Laboratories Cambridge

[3] assembled by Sam Roweis

significance α, being the null-hypothesis that the two samples are taken from populations with equal mean.

3.2. A comparison between K-means, RP K-means and IRP K-means

We start by comparing results for Algorithm 1 using $D = 10, 20, 50, 100$ and for Algorithm 2 using $D = [10, 20], [10, 20, 50], [10, 20, 50, 100]$.

On the AT&T Faces dataset (see Table 1), $IRP_{10,20,50,100}$ achieved the best average result (6.69×10^6), which compares favorably to RP_{100} (7.01×10^6) and K-Means on the original data (6.88×10^6) ($\alpha = 0.001$). $IRP_{10,20,50,100}$ (0.5s) was faster than K-Means on the original data (3.07s) ($\alpha = 0.001$).

On the NIPS 1-12 dataset (see Table 1), $IRP_{10,20,50,100}$ achieved the same result regarding MSE (4.97×10^{-3}) as K-Means on the original data. This result compares favorably to RP_{100} (4.99×10^{-3}) ($\alpha = 0.001$). $IRP_{10,20,50,100}$ (3.07 s) was faster than K-Means on the original data (59.57 s) ($\alpha = 0.001$).

On the Gaussian dataset (see Table 1), $IRP_{10,20,50,100}$ ($6.93\ \times10^3$) performed better than K-means on the original data ($8.12\ \times10^3$) and RP (for $D = 10, 20, 50, 100$). $IRP_{10,20,50,100}$ (1.10 s) was faster than K-Means on the original data (4.02 s) ($\alpha = 0.001$).

On the Uniform dataset (see Table 1), $IRP_{10,20,50,100}$ ($4.46\ \times10^3$) performed better than RP K-means (for $D = 10, 20, 50, 100$) and K-means on the original data ($4.94\ \times10^3$) ($\alpha = 0.001$). $IRP_{10,20,50,100}$ (1.10s) was faster than K-means on the original data (4.01 s) ($\alpha = 0.001$).

$IRP_{10,20,50,100}$ performed better than RP_{100} on all datasets $\alpha = 0.001$). On the AT&T Faces, Gaussian and Uniform, $IRP_{10,20}$ performed better than RP_{20}, $IRP_{10,20,50}$ better than RP_{50} and $IRP_{10,20,50,100}$ better than RP_{100} ($\alpha = 0.001$). IRP was slower than RP (for the respective dimensions) in all datasets ($\alpha = 0.001$).

The number of iterations in each dimension is shown in Table 2. It is worth mentioning that $IRP_{10,20,50,100}$ in dimension 100 has less iterations on average than RP_{100} for all datasets ($\alpha = 0.001$). This difference is explained by the fact that $IRP_{10,20,50,100}$ starts in dimension 100 using the cluster membership of dimension 50 and RP_{100} starts with a random cluster membership, as stated in Algorithms 1 and 2. $IRP_{10,20,50,100}$ in dimension 100 also has less iterations on average than K-means on the original data for all datasets ($\alpha = 0.05$) .

Table 1: MSE and running time for the several datasets. Sample average and standard deviation over 20 runs.

		$Classic$	RP_{10}	RP_{20}	RP_{50}	RP_{100}	$IRP_{10,20}$	$IRP_{10,20,50}$	$IRP_{10,20,50,100}$
AT&T F.	MSE	6.88 ±	8.64±	7.77±	7.34±	7.01±	7.53±	6.89±	6.69±
	(10^6)	0.101	0.223	0.171	0.179	0.168	0.157	0.101	0.093
	Time	3.08 ±	0.08±	0.10±	0.12±	0.19±	0.19±	0.34±	0.50±
	(s)	0.538	0.009	0.010	0.016	0.015	0.030	0.132	0.059
NIPS 1-12	MSE	4.97 ±	5.18±	5.11±	5.03±	4.99±	5.12±	5.02±	4.97±
	(10^{-3})	0.036	0.032	0.027	0.017	0.012	0.030	0.017	0.008
	Time	59.97±	0.37±	0.46±	0.83±	1.58±	0.93±	1.70±	3.07±
	(s)	21.319	0.048	0.036	0.065	0.254	0.243	0.129	0.171
Gaussian	MSE	8.12 ±	8.02±	7.62±	7.85±	8.21±	7.04±	6.97±	6.93±
	(10^3)	0.376	0.479	0.415	0.454	0.646	0.345	0.266	0.287
	Time	4.02 ±	0.16±	0.18±	0.30±	0.50±	0.32±	0.61±	1.09±
	(s)	0.932	0.013	0.021	0.008	0.021	0.021	0.033	0.035
Uniform	MSE	4.94 ±	5.61±	4.79±	4.79±	4.85±	4.65±	4.46±	4.46±
	(10^3)	0.146	0.150	0.134	0.190	0.140	0.095	0.098	0.102
	Time	4.01 ±	0.18±	0.19±	0.31±	0.51±	0.35±	0.63±	1.10±
	(s)	0.741	0.025	0.014	0.041	0.024	0.024	0.038	0.024

3.3. Choosing D for IRP-Kmeans

In this section we explore how to choose D for IRP-Kmeans. The relation $D_1 \leq D_2 \leq ... \leq D_l < d$ is heuristic and analogous to cooling in simulated annealing, where the temperature is gradually reduced, throughout this section we always consider D such that this relation is true.

The lower the dimension, the greater the probability of wrongly assigning a point to a cluster who is not actually the closest according to Euclidean distance, as in a higher temperature. This effect is explained by Theorem 2.1, which states that the distortion is within a maximum range. The actual distortion for a particular data distribution might be smaller.

In IRP K-means the lower the dimension the higher the probability of

Table 2: Iterations in each dimension for the several datasets. Sample average and standard deviation over 20 runs.

	Dim	*Classic*	*RP*	*IRP*
AT&T Faces	10	-	8.4 ± 1.59	9.2 ± 1.59
	20	-	8.6 ± 1.56	8.5 ± 2.18
	50	-	7.4 ± 1.59	6.2 ± 2.06
	100	-	7.8 ± 2.28	4.6 ± 1.40
	10304	7.1 ± 1.40	-	-
NIPS 1-12	10	-	42.3 ± 14.46	36.4 ± 13.02
	20	-	34.6 ± 8.78	37.0 ± 12.27
	50	-	31.3 ± 9.81	31.0 ± 10.20
	100	-	28.9 ± 9.10	20.0 ± 5.48
	14306	25.9 ± 9.35	-	-
Gaussian	10	-	18.7 ± 4.45	19.1 ± 5.74
	20	-	10.2 ± 3.33	7.6 ± 4.44
	50	-	5.2 ± 1.18	2.1 ± 1.25
	100	-	4.0 ± 0.79	1.65 ± 1.04
	10000	3.6 ± 1.10	-	-
Uniform	10	-	24.0 ± 7.91	23.5 ± 6.68
	20	-	15.8 ± 3.42	14.4 ± 6.68
	50	-	7.45 ± 2.26	4.15 ± 2.50
	100	-	5.45 ± 1.19	1.45 ± 1.14
	10000	3.6 ± 0.88	-	-

incorrectly assigning a point to a cluster center. The probability of assigning a point to a cluster which is not actually the closest on the original space also depends on K, as the higher the number of centers the more likely it becomes that the points are incorrectly assigned. So both these effects should be taken into account when trying to choose D optimally.

We empirically estimate this probability for each dataset by selecting randomly K points and assigning the remaining to the closest of the K. The average of each set of points defines a centroid in the original space. We then project both the centroids and all the points for a given dimension D_h, in dimension D_h we assign each of the points to one of the K centers and measure the number of points which were incorrectly assigned relatively to how they would be assigned in the original dimension. We show the average

percentage of misassigned points over 10 independent runs for a varying number of dimensions $D_h \in \lfloor 10^{0,0.1,0.2,\ldots,4} \rceil$ for the several datasets in Figure 1. For a given D_h the probability m_h of assigning a point to a cluster which is not closest on the original space varies among the datasets which suggests that the optimal D for a given l or for a maximum computational cost might be different.

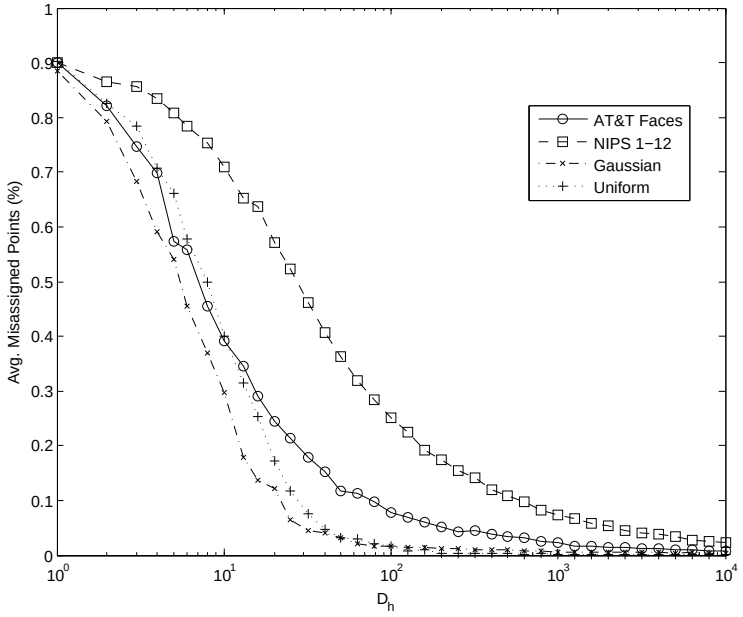


Figure 1: Average percentage of misassigned points for the several datasets. AT&T Faces ($K = 40$), NIPS 1-12 ($K = 10$), Gaussian ($K = 20$) and Uniform ($K = 20$).

If m_h is sufficiently small then we can stop at dimension D_h. Another criterion for not evaluating further dimensions is that the number of iterations in the previous dimension is the minimum, meaning that IRP has already converged.

Using this probability we empirically choose for each dataset $D_{h_{1,\ldots,l}}$ such that $m_{h_{1,\ldots,l}}$ is the closest possible to a given error. We use an exponential decay of $m_{h_{1,\ldots,l}}$ given by $m_{h_1} e^{-\lambda(h-1)}$, being m_{h_1} the initial probability of error and λ the steepness of the decay. We choose D with $l = 20$, $m_{h_1} = \frac{1}{2}$ and $\lambda = \frac{1}{8}$ for all datasets.

For each dataset we performed 20 independent runs with $l = 20$ and report the results from $l = 1$ till $l = 20$. The MSE for the several datasets is

shown in Figure 2, and the number of iterations in each dimension is shown in Figure 3. The MSE and running time for the different values of l for all datasets is shown in Table 3.

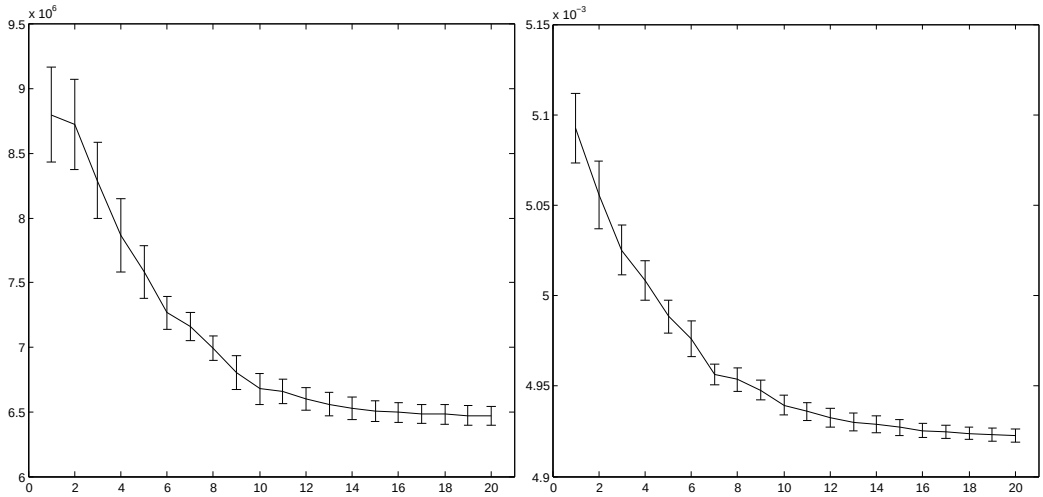
The MSE on the original space decreases gradually as the dimension increases for all datasets (see Figure 2). For both Gaussian and Uniform datasets, on which the optimal solution is known, IRP (see Table 3) approximately reaches the optimal solution MSE (6.61×10^3 and 4.39×10^3 respectively, see Section 3.1)

The number of iterations also decreases gradually as the dimension increases for all datasets (see Figure 3), being close to the minimum of possible iterations for the AT&T Faces, Gaussian and Uniform datasets. The number of iterations in dimension $l = 10$ is smaller than at $l = 5$ and at $l = 20$ less than at $l = 10$ for all datasets ($\alpha = 0.005$)

IRP for $l = 10, 15, 20$ (see Table 3) reaches a lower MSE than K-means on original data while being faster with $l = 10, 15$ for all datasets ($\alpha = 0.005$).

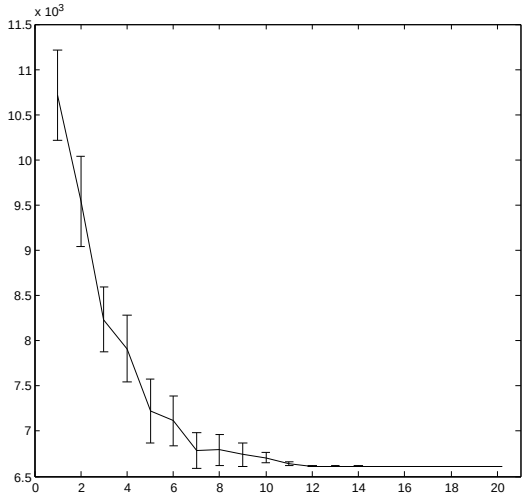
The good results across several datasets with distinct data distributions (see Figure 1) indicate that the optimal D should result in a decay on the probability of assigning a point to a cluster which is not the closest on the original space, for which m_h is an empirical approximation.

Finally to illustrate the differences in the solutions between K-means and IRP K-means we generate one last synthetic dataset with $K = 10$ and 1000 dimensions. We visualize the solutions of each method using Principal Components Analysis (PCA). Each cluster j is sampled from a multivariate Uniform distribution $\mathcal{U}(c_j - d_j, c_j + d_j)$ where c_j is sampled from a multivariate Uniform distribution $\mathcal{U}(-1, 1)$ and d_j is sampled from a multivariate Uniform distribution $\mathcal{U}(0, 2)$. The dataset contains 1000 points randomly sampled from each cluster with probability $1/K$. The MSE of assigning all points to the correct cluster is 4.23×10^2 for this dataset. We perform 10 independent runs of K-means and $IRP_{l=20}$ using m_h to choose D_h as before. The average MSE of $IRP_{l=20}$ is also 4.23×10^2 reaching always the same solution. The average MSE of K-means is 4.70×10^2. After we obtained the clusters, we use PCA to represent the data in 2 dimensions and show the cluster centers closest to the average MSE of each method on Figure 4. It can be seen that IRP clearly discovered the original clusters while K-means in some cases has more than a center covering a cluster while on others the centers are in between the actual clusters.

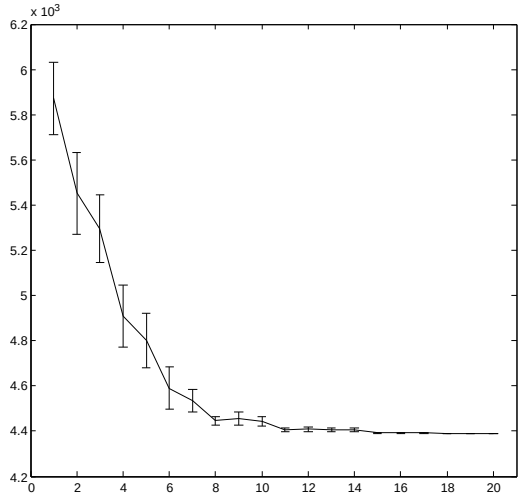


(a) AT&T Faces

(b) NIPS 1-12



(c) Gaussian



(d) Uniform

Figure 2: MSE (vertical axis) for different values of l (horizontal axis) for the several datasets. Sample average and standard deviation over 20 runs.

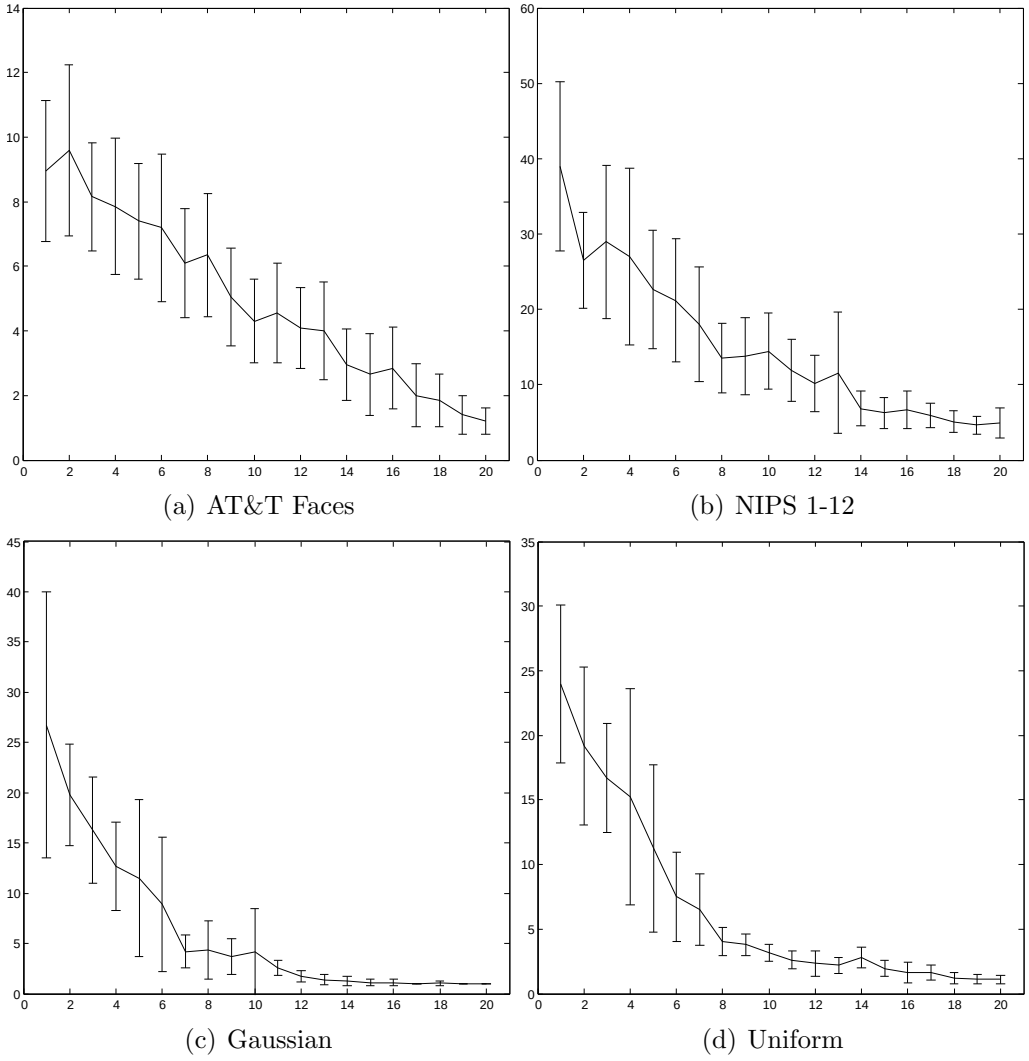
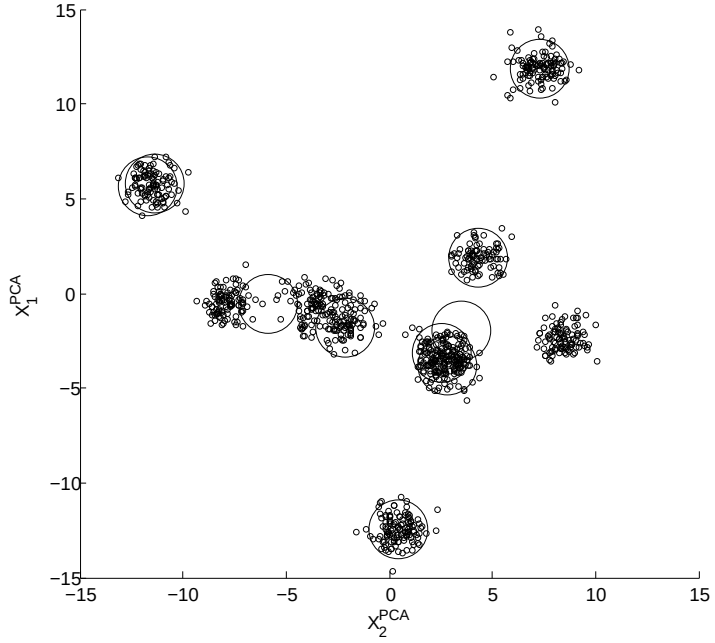


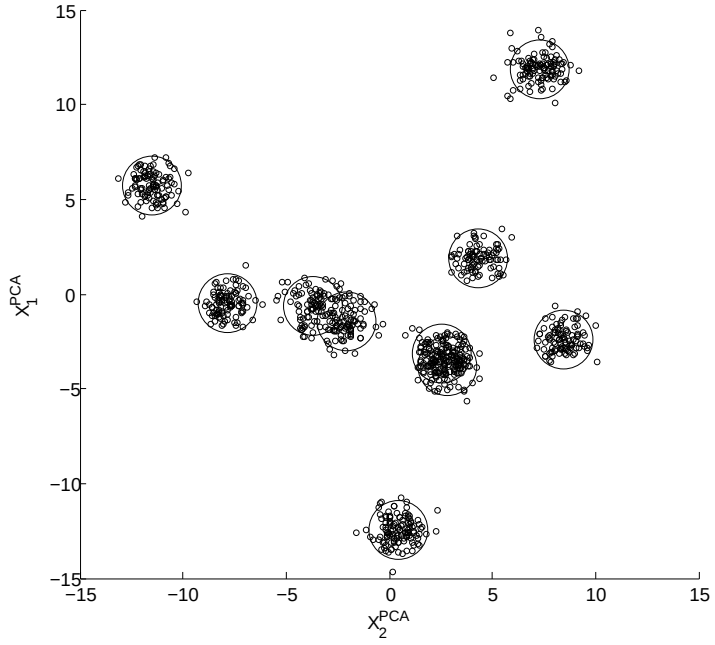
Figure 3: Iterations (vertical axis) at dimension $D_{h=l}$ (horizontal axis) for the several datasets. Sample average and standard deviation over 20 runs.

Table 3: MSE and running time for the several datasets using IRP ($l = 5, 7, 10, 15, 20$). Sample average and standard deviation over 20 runs.

		$Classic$	$IRP_{l=5}$	$IRP_{l=7}$	$IRP_{l=10}$	$IRP_{l=15}$	$IRP_{l=20}$
AT&T F.	MSE	$6.88 \pm$	$7.58 \pm$	$7.16 \pm$	$6.68 \pm$	$6.51 \pm$	$6.47 \pm$
	(10^6)	0.101	0.203	0.109	0.120	0.080	0.074
	Time	$3.08 \pm$	$0.30 \pm$	$0.42 \pm$	$0.63 \pm$	$1.15 \pm$	$2.26 \pm$
	(s)	0.538	0.041	0.048	0.064	0.085	0.088
NIPS 1-12	MSE	$4.97 \pm$	$4.99 \pm$	$4.96 \pm$	$4.94 \pm$	$4.93 \pm$	$4.92 \pm$
	(10^{-3})	0.036	0.009	0.006	0.006	0.004	0.004
	Time	$59.97 \pm$	$3.43 \pm$	$6.07 \pm$	$11.99 \pm$	$38.32 \pm$	$118.32 \pm$
	(s)	21.319	0.207	0.291	0.345	0.595	1.376
Gaussian	MSE	$8.12 \pm$	$7.22 \pm$	$6.78 \pm$	$6.71 \pm$	$6.61 \pm$	$6.61 \pm$
	(10^3)	0.376	0.358	0.201	0.056	0.001	0.000
	Time	$4.02 \pm$	$0.67 \pm$	$0.92 \pm$	$1.30 \pm$	$2.01 \pm$	$2.90 \pm$
	(s)	0.932	0.062	0.062	0.073	0.072	0.092
Uniform	MSE	$4.94 \pm$	$4.80 \pm$	$4.53 \pm$	$4.44 \pm$	$4.39 \pm$	$4.39 \pm$
	(10^3)	0.146	0.119	0.049	0.020	0.003	0.001
	Time	$4.01 \pm$	$0.76 \pm$	$1.04 \pm$	$1.48 \pm$	$2.37 \pm$	$3.43 \pm$
	(s)	0.741	0.047	0.046	0.051	0.058	0.106



(a) K-means



(b) IRP K-means

Figure 4: Comparison between K-means and IRP K-means using PCA. Each point is represent by a small circle while each center is represented by a big circle.

4. Discussion

We introduced IRP K-means (see Section 2.2), a method for clustering high-dimensional data which builds on random projection and K-means. The iterative increase of dimensionality is analogous to cooling in simulated annealing K-means [12], whose purpose is to avoid local minimums. However unlike simulated annealing clustering it can reduce the running time instead of increasing it orders of magnitude [2].

Experimental results on four high-dimensional datasets showed good results on an image dataset, a text dataset, and two synthetic datasets. IRP K-means compared favorably to RP K-means (see Section 3.2), IRP with $(D = 10, 20, 50, 100)$ achieved a lower MSE than RP $(D = 100)$ on all datasets $(\alpha = 0.001)$.

Experimental results indicate that a good criterion for choosing D is (see Section 3.3) according to a gradual decrease in the probability m_h of a assigning a point to the a cluster which is not the closest in the original space. Choosing D according to this criterion, IRP (with $l = 10, 15, 20$) reaches a lower MSE than K-means on original data for all datasets while being faster thank K-means (with $l = 10, 15$) $(\alpha = 0.005)$.

IRP K-means can be used with other clustering algorithms like a K-means approximation algorithm [8]. Algorithm 2 for $D = [D_1, D_2, ..., D_l]$ is equivalent to initializing Algorithm 1 (with $D = D_l$) with cluster membership obtained from Algorithm 2 (with $D = [D_1, D_2, ..., D_{l-1}]$). Therefore we can use the algorithm described in [3] for D_l, and hold its MSE guarantees since dimensions $1, 2, ..., l-1$ would be used only to initialize it.

Acknowledgments. The authors would like to thank João Sacramento and two anonymous reviewers for their much helpful comments. This work was supported by Fundação para a Ciência e Tecnologia (INESC-ID multiannual funding) through the PIDDAC Program funds and through an individual doctoral grant awarded to the first author (contract SFRH/BD/61513/2009).

[1] Achlioptas, D., 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. Journal of Computer and System Sciences 66, 671 – 687. Special Issue on PODS 2001.

[2] Al-Sultan, K.S., Khan, M.M., 1996. Computational experience on four algorithms for the hard clustering problem. Pattern Recognition Letters 17, 295 – 308.

[3] Boutsidis, C., Zouzias, A., Drineas, P., 2010. Random Projections for k-means Clustering, in: Advances in Neural Information Processing Systems 23, pp. 298 – 306.

[4] Dasgupta, S., 2000. Experiments with random projection, in: Uncertainty in Artificial Intelligence: Proceedings of the Sixteenth Conference (UAI-2000), pp. 143 – 151.

[5] Dasgupta, S., Gupta, A., 2003. An elementary proof of a theorem of johnson and lindenstrauss. Random Structures & Algorithms 22, 60–65.

[6] Hecht-Nielsen, R., 1994. Context vectors: general purpose approximate meaning representations self-organized from raw data. Computational Intelligence: Imitating Life , 43–56.

[7] Johnson, W., Lindenstrauss, J., 1984. Extensions of lipschitz mappings into a hilbert space. Contemp. Math. 26, 189–206.

[8] Kumar, A., Sabharwal, Y., Sen, S., 2004. A simple linear time $(1 + \epsilon)$-approximation algorithm for k-means clustering in any dimensions, in: Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on, pp. 454 – 462.

[9] Li, P., Hastie, T.J., Church, K.W., 2006. Very sparse random projections, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA. pp. 287–296.

[10] Lloyd, S., 1982. Least squares quantization in pcm. Information Theory, IEEE Transactions on 28, 129 – 137.

[11] Magen, A., 2002. Dimensionality reductions that preserve volumes and distance to affine spaces, and their algorithmic applications, in: Randomization and Approximation Techniques in Computer Science. Springer. volume 2483 of *Lecture Notes in Computer Science*, pp. 953–953.

[12] Selim, S.Z., Alsultan, K., 1991. A simulated annealing algorithm for the clustering problem. Pattern Recognition 24, 1003 – 1008.