PCA-guided search for K-means

Report for the class Geometric Data Analysis

Timothé Boulet
Thomas Lemercier
Master MVA
France

ABSTRACT

In this report, we present the PCA-guided search [Xu et al. 2015], which aims to enhance the K-means clustering algorithm, one of the most widely employed techniques in data analysis. Despite its popularity, K-means often struggles with converging to a global solution due to its non-convex formulation and the propensity to get trapped in local minima. To tackle this challenge, the proposed method finds a relevant initilization point, by leveraging the fact that the optimal global solution for a relaxed version of K-means clustering lies exactly within the PCA subspace. It is thus possible to start from there before running the actual K-means algorithm. Additionnally, we demonstrate the empirical relevance of this initialization method, by comparing it against modern heuristics and other initialization techniques.

1 INTRODUCTION

Data clustering, also referred to as unsupervised classification, is a crucial method in machine learning and data analysis, aimed at grouping objects into clusters based on their similarities. This approach finds applications in diverse fields, including exploratory pattern analysis, data mining, decision-making, machine learning, vector quantization, and compression scenarios.

As previously mentioned, among the various clustering algorithms, the K-means algorithm stands out for its simplicity, efficiency, and parallelizability, rendering it widely employed for addressing practical problems, on parallel and distributed computing platforms.

However, the K-means algorithm, faces challenges in converging to a global solution due to the non-convex nature of its formulation. Local minima near the initialization can hinder its effectiveness, especially in high-dimensional data where there are exponentially many local solutions.

To address these challenges, two main directions have been explored:

(1) Methods to escape local minima, often employing meta-heuristics such as simulated annealing, genetic algorithms
(2) Designing better initialization algorithms

The paper we have been working on focuses on the second approach, specifically exploring the relationship between K-means clustering and principal component analysis (PCA). Recent theoretical analyses have revealed that the global solution to K-means clustering lies in the PCA subspace. The authors present a method named PCA-guided search for K-means clustering, achieving a more effective exploration within the PCA subspace. Subsequently, by utilizing the solution obtained in the PCA subspace, the original space is considered.

Leveraging the computational efficiency of K-means clustering in low-dimensional spaces, the proposed approach aims to achieve better solutions with reduced computational complexity.

2 THE PCA AND K-MEANS RELATION

In a previous study [Ding and He 2004], the authors explored the connection between Principal Component Analysis (PCA) and the K-means algorithm. This section discusses two key findings that laid the groundwork for the algorithm we'll be covering. We'll break down the relationship between PCA and K-means, looking into how PCA's smooth solutions and K-means' group memberships interact. These findings are the basis for the PCA-guided search algorithm, and we'll go over its development and analysis in the next sections.

2.1 Solution of K-means clustering lies in PCA-subspace

First, let's introduce common notations for both the PCA and K-means problems.

PCA consists in computing the k principal eigenvectors $U = (u_1, ..., u_k)$ of the covariance matrix $Cov(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\top$ of the data $X \in \mathbb{R}^{p\times n}$ where $\bar{x}$ denotes the mean of X. Then, the solution of the PCA problem is $Y = (y_1, ..., y_n)$, with

$$y_i = U^\top x_i. \tag{1}$$

For this solution, we have by construction that $Cov(Y)$ is diagonal, and we can define the whitened solution by defining $\tilde{Y} = (\tilde{y}_i, ..., \tilde{y}_n)$ with $\tilde{y}_i = (U\Sigma^{-\frac{1}{2}})^\top x_i$ where $\Sigma = diag(\lambda_1, ..., \lambda_k)$ and $(\lambda_1, ..., \lambda_k)$ are the eigenvalues associated to U.

K-means clustering aims at minimizing the function

$$J = \sum_{j=1}^{K}\sum_{x_i\in C_j} \|x_i - \mu_j\|^2 \tag{2}$$

where μ_j is the centroid of cluster C_j, and the norm is Euclidean. The solution for the hard clustering problem is a binary indicator matrix $H = (h_1, ..., h_K) \in \mathbb{R}^{n\times K}$, where

$$H_{ij} = \begin{cases} \frac{1}{\sqrt{\#C_j}} & \text{if } x_i \text{belongs to cluster } C_j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $\#C_j$ denotes the cardinal of cluster C_j. The relaxed clustering problem aims at minimizing J, without the constraint of having a binary H, and allowing attribution to clusters in the form of $\tilde{H} \in (-1, 1)$, due to rotations we introduce right bellow.

Theorem 2.1. *Using previous notations, $\tilde{H}^* = \tilde{Y}^\top R^\top$, where $\tilde{H}^*$ is the optimal relaxed indicator matrix, and $R \in \mathbb{R}^{K\times K}$ is an orthogonal rotation matrix.*

Proof. By definition, the centroid of each cluster is given by $\mu_j = \frac{1}{n_j}\sum_{x_i \in C_j} x_i = \frac{1}{\sqrt{n_j}} X h_j$. We can thus rewrite the objective function (also know as the distortion J):

$$\begin{aligned}
J &= \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2 \\
&= \sum_{j=1}^{K} \sum_{x_i \in C_j} \left(\|x_i\|^2 - 2\mu_j^\top + \|\mu_j\|^2 \right) \\
&= \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i\|^2 - \sum_{j=1}^{K} \mu_j^\top \sum_{x_i \in C_j} \left(2x_i - \mu_j\right) \\
&= \sum_{i=1}^{n} \|x_i\|^2 - \sum_{j=1}^{K} n_j \mu_j^\top \mu_j \\
&= \sum_{i=1}^{n} \|x_i\|^2 - \underbrace{\sum_{j=1}^{K} h_j^\top X^\top X h_j}_{tr(H^\top X^\top X H)}
\end{aligned}$$

Thus, $\min J = \max tr(H^\top X^\top X H) = \max tr(\tilde{H}^\top X^\top X \tilde{H})$, where $\tilde{H} = HR$, and R is any orthogonal rotation matrix. Besides, we have $H^\top H = I = \tilde{H}^\top \tilde{H}$ due to the orthogonality of clustering membership. Now, the optimization problem has become:

$$\max_{\tilde{H}} \quad tr(\tilde{H}^\top X^\top X \tilde{H}) \text{ such that } \tilde{H}^\top \tilde{H} = I \tag{4}$$

which is a classic discrete optimization problem known to be NP-hard. The optimal solution $\tilde{H}^*$ of the relaxed version (obtained by allowing H to have values in $(0, 1)$) is the K eigenvectors $V = (v_1, \ldots, v_K)$ of the Gram matrix $X^\top X$ associated with the K largest eigenvalues $(\xi_1, \ldots, \xi_K)$: $\tilde{H}^* = V$, i.e., $H^* = VR^T$. Finally, we can make use of the singular value decomposition (SVD) of X:

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^\top = U \Sigma V^\top, \tag{5}$$

with r the rank of X, to obtain that $\tilde{Y} = (U\Sigma^{-1})^\top X = V^\top = \tilde{H}^{*\top} = R^\top \tilde{H}^{*\top}$, and that $\sigma_k = \sqrt{\lambda_k} = \sqrt{\xi_k}$. This proves that $\tilde{H}^* = \tilde{Y}^\top R$ □

Definition 2.2. The optimal centroids $M = (\mu_1, \ldots, \mu_K)$ for K-means span a subspace called *cluster centroid subspace*. Conversly, the principle directions of PCA $U = (u_1, \ldots, u_k)$ span the *PCA-subspace*.

Theorem 2.3. *Cluster centroid subspace and PCA subspace are identical.*

Proof. Using previous notations, we start by constructing an orthogonal basis of the cluster centroid subspace $\tilde{M} = M(M^\top M)^{-1/2}$. Therefore, the projection of any data point x_i on the cluster centroid subspace is given by $\tilde{M}\tilde{M}^\top x_i$. Now, we recall that the centroid of each cluster is written $\mu_j = \frac{1}{\sqrt{n_j}} X h_j$, hence $M = XHN^{-1/2}$ with $N = diag(n_1, \ldots, n_K)$. It follows that:

$$\begin{aligned}
\tilde{M}\tilde{M}^\top &= M(M^\top M)^{-1} M^\top \\
&= XHN^{-1/2}(N^{-1/2}H^\top X^\top X H N^{-1/2})^{-1}(XHN^{-1/2})^\top \\
&= XH(H^\top X^\top X H)^{-1}(XH)^\top \\
&= X\tilde{H}(\tilde{H}^\top X^\top X \tilde{H})^{-1}(X\tilde{H})^\top.
\end{aligned}$$

Thanks to Theorem 2.1, we have that the solution of K-means clustering is $(h_1, \ldots, h_K)R = V$, i.e., $\tilde{H} = V$. Thus we have

$$\tilde{M}\tilde{M}^\top = XV(V^\top X^\top X V)^{-1}(XV)^\top. \tag{6}$$

From 5, we obtain that: $Xv_i = \sigma_i u_i$, i.e., $XV = U\Sigma$. Hence

$$\tilde{M}\tilde{M}^T = U\Sigma(\Sigma^\top U^\top U \Sigma)^{-1}(U\Sigma)^\top = UU^\top, \tag{7}$$

which means that the cluster centroid subspace is equal to the PCA subspace. □

Therefore, the optimal solution of the relaxed K-means clustering problem lies in the k-dimensional PCA subspace, which is smaller than the original space. Using such values for the initialization should allow the non-relaxed K-means to converge more quickly to a global optimum, with less probability to be stuck in a local optium located far away from a global one.

2.2 PCA-guided search for K-means

Based on the theoretical properties of PCA and K-means that were just highlighted, the authors of the paper proposed the PCA-Guided Kmeans algorithm[1].

3 OTHER CLUSTERING ALGORITHMS

To assess the proposed method, the authors conducted a comparison with various clustering techniques. The methods under comparison encompass KMeans++, classical K-means with random initialization, KR, KKZ, HAC, and PCA-part that we summarize in explain in the following.

3.1 Random Initialization

The authors distinguish between two random initialization methods, R1 and R2. R1 consist to randomly partition the dataset in k random clusters. Those became our initial clusters (from which we deduce the centroids). R2 consist of randomly taking k points of the dataset. Those became our initial centroids.

The authors conjecture that R2 is better because in R1, every initial centroid will be roughly at the center of the dataset, which hinders the convergence process. However, as highlighted by the paper's authors, there are studies [J.M. Pena * 1999] which suggest method R1 gives better results than method R2. For confirming the authors hypothesis, we include both methods in our benchmark.

3.2 K-Means++

K-means++ adopts a greedy approach in selecting K centroids. The first centroid is chosen randomly, and subsequent centroids are selected with a probability proportional to the square of the shortest distance from a data point to the closest centroid.

Algorithm 1 PCA-Guided K-Means

Inputs

(1) A dataset of points $D_n = (X_i)_{i\in\{1,\cdots,n\}} \in \mathbb{R}^d$
(2) $k \leq n$, the number of clusters

Dimension reduction

Apply PCA reduction to the data, setting the dimension of the reduction to k:

$$\tilde{D}_n = \text{PCA}(D_n)$$

Clustering on $\tilde{D}_n$

Apply the K-means algorithm to the dimensionally reduced dataset, obtaining a binary indicator matrix $H \in \mathbb{R}^{n\times k}$ such that $H_{i,j} = 1$ if point i is assigned to cluster j and 0 otherwise:

$$H = \text{K-Means}(\tilde{D}_n, \text{R2(Random Initialization)})$$

Computing the initial centroids in the original space

Using the binary indicator matrix H, compute the centroids $(\mu_1, \cdots, \mu_k)$ in the original subspace:

$$\mu_j = \frac{\sum_{i=1}^{n} H_{i,j} X_i}{\sum_{i=1}^{n} H_{i,j}}$$

Clustering on D_n

Using the previously obtained centroids $(\mu_1, \cdots, \mu_k)$ as the initial centroids, perform the K-Means algorithm:

$$H = \text{K-Means}(D_n, (\mu_1, \cdots, \mu_k))$$

3.3 KKZ and "KKZ-KMeans"

The KKZ algorithm initiates by selecting the first centroid with the maximum norm. Successive centroids are chosen based on the maximal distance from previously selected centroids to candidate points, iteratively repeating this process until K centroids are identified. The algorithm "KKZ-KMeans" is simply using KKZ as an centroid initialization algorithm, and then applying a KMeans algorithm for convergence.

3.4 KR Initialization

The KR algorithm sequentially chooses centroids, with the first centroid selected as the most centrally located in the dataset. Subsequent centroids are then chosen to be far away from the previously selected ones while still having many data points close to them. As the algorithm was not clearly identified, we opted to exclude it from our experiment.

3.5 HAC and "HAC-KMeans"

Several studies utilize the results of hierarchical agglomerative clustering (HAC) as centroid initialization. HAC, employing a "bottom-up" approach, begins with numerous small clusters and iteratively merges the closest clusters until a single cluster is formed. The algorithm "HAC KMeans" is simply HAC with and additional KMeans algorithm applied after it for convergence.

3.6 PCA-part

The PCA-part method involves recursively partitioning a current cluster into two by computing the first principal component. The sign of each element is used to split the cluster, and this process is repeated until K clusters are obtained. Because of a lack of understanding, this algorithm was not included in our benchmark.

3.7 Our benchmarked algorithms

These varied initialization strategies provide diverse approaches to addressing the challenge of clustering. However for our experiment we decided to focus on the following algorithm: R1 and R2 initialization, KKZ algorithm, KKZ-KMeans, HAC clustering, HAC-Kmeans clustering, Kmeans++, and PCA-guided search.

4 EXPERIMENTAL SETUP

We detail here the exact conditions of our experiments.

4.1 Datasets description

We performed experiments on five distinct datasets selected from the classical clustering or classification datasets found in the literature. In an effort to replicate the findings of the original paper, we incorporated the same four datasets utilized for their benchmark in the study and we also included the IRIS dataset as well.

The first four datasets are all datasets of images representing images belonging to a certain class, and the last one being a classical tabular dataset. The class of an image represents it's cluster, which allows us to specify in advance the number of clusters k for our K-means algorithm, e.g. 10 clusters for MNIST.

The datasets are the following :

- **AT&T** : The AT&T Face dataset comprises ten images for 40 different individuals (k=40), each with a size of 92 × 112 pixels and 256 grey levels per pixel. In our study, we consider pixels as features. The images were resized to 23 × 28, transformed into vectors of dimension 644 by concatenating the matrix rows.
- **MNIST** : The MNIST Handwritten Digits dataset includes 8-bit grayscale images representing digits "0" through "9," (k=10) with around 7000 examples for each digit. We flattened our images to dimension 784.
- **Binary Alphabet** : The Binary Alphabet dataset consists of 26 handwritten alphabets "A" to "Z," (k=26) with 39 examples for each letter. Each image is a 20 × 16 binary image, converted into vectors of dimension 320.
- **Coil20** : We used 360 images of 5 items (k=5), from the Coil20 dataset, which consist of grayscale images of objects. Each image is 32 × 32, and we convert them into vectors of dimension 1024.
- **IRIS** : The Iris dataset consists of 150 samples of iris flowers, each belonging to one of three species: setosa, versicolor, or virginica (k=3). This is a very simple clustering dataset. Each sample has four features: sepal length, sepal width, petal length, and petal width, all measured in centimeters. The dataset is often used for classification tasks in machine learning.

4.2 Metrics

We used the classical metric used for clustering, i.e. distortion :

$$J = \sum_{j=1}^{K} \sum_{x_i \in C_j} \|x_i - \mu_j\|^2$$

This is the average L_2 distance between a data point and the centroid of its cluster assigned by the clustering algorithm.

Specifically, in each run, we execute multiple iterations of the algorithm and record the best distortion achieved since the start of the run. This metric is referred to as "best distortion" or "best distortion over time":

$$J_{\text{best}}(t) = \max_{0 \le l \le t} J(l)$$

with $J(l)$ being the distortion obtained for the l-th run.

This metric signifies the practical performance of our algorithms and mirrors the outcomes that the algorithms would manifest in real-world scenarios. The original paper overlooked this metric, as it solely showcased the distortion in decreasing order, failing to accurately depict the results one would encounter when deploying those algorithms in practical applications.

4.3 Experimental settings

For every algorithm and dataset, we conducted multiple runs, each comprising 1000 iterations of our algorithm. However, in the exceptional case of MNIST, which had a significantly larger dimension compared to other datasets, we limited the iterations to 100. If, for a specific dataset, all algorithms converged after a certain number of iterations, we only showcase those initial iterations for the sake of clarity.

For stochastic algorithms (Random, KMeans++ and PCA Guided Search), we ran 10 runs with different seeds to measure the empirical mean and empirical standard deviation of our metrics. We plot the mean and the standard deviation, represented by the shaded envelope around the mean.

For deterministic algorithm (KKZ and HAC), we performed only one run and plotted the result of the first iteration.

The implementation of our algorithms, in particular of K-Means, are done with the scikit-learn python library, or by ourselves when the algorithms were not implemented in it.

The code, along with an explicative README document, can be found in our GitHub repository.

5 RESULTS

We plot here the obtained results. They can be found on our Weight and Biases project. We advice the reader to group runs by "dataset_name" and "algo_name" and to display runs dataset by dataset.

5.1 Complementary analysis

We plot in this section $J_{\text{best}}(t)$ the results of the best distortion obtained among previous iterations, in function of t the current number of iterations done.

For the AT&T dataset[1], we observed that HAC is the most effective clustering algorithm, followed by K-means++, the proposed PCA-Guided Kmeans algorithm, and, among the two traditional Kmeans initialization schemes, R2 yielded superior results.

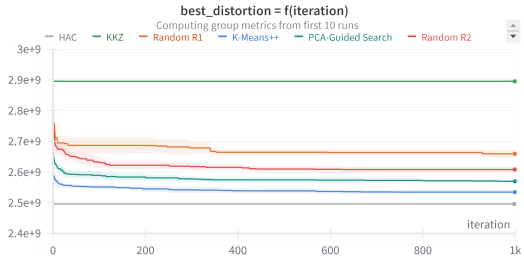

Figure 1: Results on AT&T

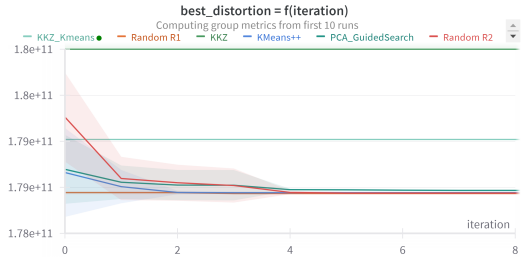

Figure 2: Results on MNIST

In the case of the MNIST dataset[2], all algorithms demonstrated comparable performance.

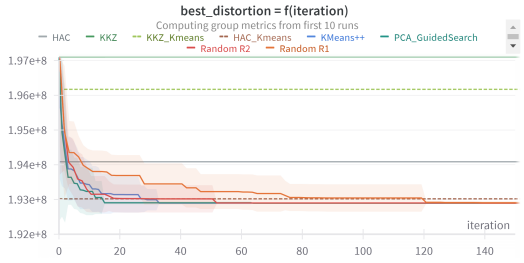

Figure 3: Results on Coil20

Similarly, for the coil20 dataset[3], most algorithms exhibited similar levels of performance.

On the Binary alphabet dataset[4], the best performing algorithm is the proposed algorithm, follow by HAC-Kmeans, Kmeans++, R1 initilization and R2 initilization.

For clarity [5], we reported the result on the IRIS dataset in the next tables[1][2]:

Algorithm	Kmeans++	PCA-Guided	R2
Performance	78.851	78.851	78.851

Table 1: Performance on IRIS - Part 1

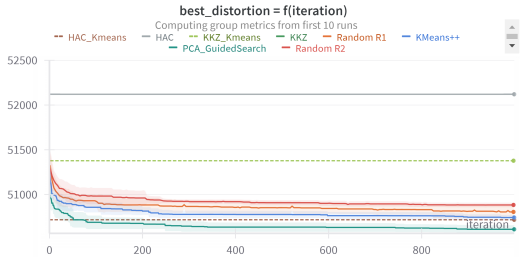

Figure 4: Results on Binary Alphabet. The KKZ was above 56000 and was not plot.

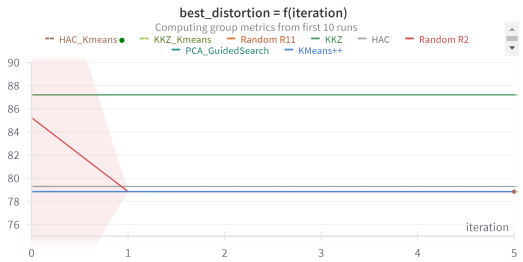

Figure 5: Results on IRIS

Algorithm	HAC	R1	KKZ_Kmeans	HAC_Kmeans
Performance	79.297	78.851	78.851	78.851

Table 2: Performance on IRIS - Part 2

The outcomes on the IRIS dataset are misleading because the clustering task is overly simplistic, leading to nearly identical performances from all algorithms in nearly every iterations.

Although clustering algorithms are highly sensitive to the characteristics of the dataset, it is obvious in our case that the top-performing algorithms are PCA-Guided Kmeans, Kmeans++, and HAC. To gain a deeper insight into these algorithms, we compile the average run time for a single run of each on every dataset.

Dataset/Algorithm	Kmeans++	PCA-Guided	HAC
AT&T	1.4s	0.8s	6s
MNIST	4.9s	4.7s	Too Long
Coil20	0.04s	0.05s	5s
Binary Alphabet	0.09s	0.07s	6s

Table 3: Time comparaison of each algorithm on each dataset

When examining this table, it's important to emphasize that it is only suitable for comparing execution times across algorithms, not datasets. This limitation arises from differences in data set dimensions and the fact that the computations were carried out on distinct computers, with AT&T and Binary Alphabet on one system and MNIST and Coil20 on another.

From a time to performance stand point, the PCA-Guided Search is clearly the winner, however it should be noted that the performance over time of the algorithm tends to increase at the same rate indeed by examining closely the curve none of them cross each others, except obvious for the HAC algorithm which is deterministic.

Additionally, it is important to highlight that contrary to the authors' assertion, the R2 initialization is not consistently superior to the R1 initialization, as evidenced by the results from the Binary Alphabet dataset.

Finally, the most surprising conclusion we draw is the disparity between our findings and the results reported in the original paper:

The PCA-Guided Search does not consistently outperform the Kmeans++ algorithm across all datasets. On most datasets we tested, it performs comparably to Kmeans++, and it only surpasses Kmeans++ on the Binary Alphabet dataset.

6 CONCLUSION

In conclusion, we have investigated the PCA-guided search algorithm for K-means clustering, exploring both theoretical aspects and on a benchmark against the typical concurrent methods. The algorithm is very promising from a theoretical point of view, the results of the benchmark are overall positive. Notably, the implementation we built based on the scikit-learn package in Python has demonstrated that the algorithm is very time efficient. However, we must point out the limits of K-means itself, which suffers from strong assumptions such as spherical cluster shapes, equal cluster sizes, and the idea that the mean is the best central tendency measure for all clusters. These hypothesis may not hold true in all datasets, potentially affecting the performances of the algorithm. The exploration of PCA-guided search for K-means opens up interesting possibilities for research, particularly in investigating whether similar dimensionality reduction techniques could be beneficially applied to other classic clustering methods, with similar theoretical insights.

REFERENCES

Chris Ding and Xiaofeng He. 2004. K-Means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-First International Conference on Machine Learning* (Banff, Alberta, Canada) *(ICML '04)*. Association for Computing Machinery, New York, NY, USA, 29. https://doi.org/10.1145/1015330.1015408

J.A. Lozano P. Larranaga J.M. Pena *, 1. 1999. An empirical comparison of four initialization methods for the K-Means algorithm. *Pattern Recognition Letters*. 20 (1999), 1027–1040.

Qin Xu, Chris Ding, Jinpei Liu, and Bin Luo. 2015. PCA-guided search for K-means. *Pattern Recognition Letters* 54 (2015), 50–55.