applied sciences

Article

Efficient High-Dimensional Kernel k-Means++ with Random Projection

Jan Y. K. Chan , Alex Po Leung * and Yunbo Xie

Faculty of Information Technology, Macau University of Science and Technology, Taipa 999078, China; janchanyk@gmail.com (J.Y.K.C.); stephenyunboxie@outlook.com (Y.X.)
* Correspondence: pleung@must.edu.mo

Abstract: Using random projection, a method to speed up both kernel k-means and centroid initialization with k-means++ is proposed. We approximate the kernel matrix and distances in a lower-dimensional space $\mathbb{R}^d$ before the kernel k-means clustering motivated by upper error bounds. With random projections, previous work on bounds for dot products and an improved bound for kernel methods are considered for kernel k-means. The complexities for both kernel k-means with Lloyd's algorithm and centroid initialization with k-means++ are known to be $O(nkD)$ and $\Theta(nkD)$, respectively, with n being the number of data points, the dimensionality of input feature vectors D and the number of clusters k. The proposed method reduces the computational complexity for the kernel computation of kernel k-means from $O(n^2D)$ to $O(n^2d)$ and the subsequent computation for k-means with Lloyd's algorithm and centroid initialization from $O(nkD)$ to $O(nkd)$. Our experiments demonstrate that the speed-up of the clustering method with reduced dimensionality $d = 200$ is 2 to 26 times with very little performance degradation (less than one percent) in general.

Keywords: kernel k-means; k-means++; random projection; dimensionality reduction; high dimensional data

check for updates

Citation: Chan, J.Y.K.; Leung, A.P.; Xie, Y. Efficient High-Dimensional Kernel k-Means++ with Random Projection. *Appl. Sci.* **2021**, *11*, 6963. https://doi.org/10.3390/app11156963

Academic Editors: Sławomir Nowaczyk, Mohamed-Rafik Bouguelia and Hadi Fanaee

Received: 21 June 2021
Accepted: 25 July 2021
Published: 28 July 2021

1. Introduction

The clustering problem is one of the oldest and most important problems in machine learning. k-means clustering has attracted increasing attention recently. It can be used in many fields, such as market segmentation, social network analysis, and astronomical data analysis. k-means clustering is an NP-hard problem. In this problem, we are given n data points,

$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \tag{1}$$

in the D-dimensional space. The goal is to separate the points $\mathbf{X}$ into k clusters $\mathbf{C}$,

$$\mathbf{C} = \{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_k\} \tag{2}$$

where $\mathbf{C}_i$ denotes the i-th cluster, so as to minimize the potential function, which is the total sum of squared distances between each point to its closest center. That is,

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{\mathbf{x} \in \mathbf{C}_i} \|\mathbf{x} - \mathbf{c}_i\|^2 \tag{3}$$

where $\mathbf{c}_i$ denotes the center point of the i-th cluster.

Data in high dimensionality has become an increasingly frequent problem in machine learning. Memory and the computational cost would be reduced if dimensionality can be reduced. To deal with this issue, plenty of techniques have been proposed, such as locally linear embedding, principle component analysis (PCA), ISOMAP, and multidimensional scaling. However, high computational training overheads and complex procedures are involved with a lot of the proposed methods, especially with very high-dimensional data. Because of this, there has been a surge of interest in new, simple and more computationally efficient techniques using random projection for very high-dimensional data, for example, [3–5].

In addition, kernel k-means is a popular method due to the fact that k-means clustering utilizes the Euclidean distance metric to calculate the similarity between data points, implying that only linearly separable clusters in the original input space can be found. Obviously, the measure is not suitable for just any dataset.

With kernel methods, data in the original input space is explicitly mapped into a different feature space through the user-specified kernel function. Features in data points are mapped to higher dimensional feature spaces. Putting data points in higher dimensional spaces after they have been mapped is a reasonable approach without the assumption that data points are linearly separable, which is also applicable for traditional k-means clustering. This advantage has made kernel k-means popular in recent years [6,7]. Kernel k-means, which is an extension of k-means, computes kernels instead of the Euclidean distance for point-wise distances. Kernel k-means projects data points into high-dimensional kernel space and performs k-means clustering in the mapped feature space to allow non-linearly separable clusters. In kernel k-means, it is required to compute an $n \times n$ kernel matrix with feature vectors in the original space, and the computation cost is a function of n and the D.

To improve the performance with centroid initialization, the k-means++ seeding algorithm [8] is used to obtain k initial centers, which give the expected $O(log\ k)$ approximation ratio, as shown by Arthur and Vassilvitskii. In [8], the k-means++ algorithm is introduced, which is a seeding algorithm for k-means to replace the uniform distribution of Lloyd's initialization with a probability distribution given by the points' squared distances from the closest chosen centers as the weights, i.e., the weight for data point $\mathbf{x}$ is $\frac{Dist(\mathbf{x})^2}{\sum_{\mathbf{x} \in \mathbf{X}} Dist(\mathbf{x})^2}$ with $Dist(\mathbf{x})$ denoting the shortest distance to the closest center already chosen. k-means++ provides a performance guarantee of only a constant factor from the optimum and is proven to be $O(log\ k)$-competitive. The performance of the k-means++ algorithm is guaranteed, but the computational effort of the clustering still depends on the number of points n, dimensionality D and the number of clusters to be discovered k with the complexity of $O(Dnk)$.

The remainder of the paper describes our contributions by applying kernel k-means with random projection and efficient initialization in the section "Our Method". In the Experiment Section, a number of experimental results have been shown. The optimal dimensionality, an ablation study, and a number of accuracy measures for our method are discussed before the Conclusion Section.

Contributions

In this work, there are two main contributions in the kernel k-means clustering method. The first contribution of this paper is the proposal of a sped-up version of k-means++. By using this centroid initialization prior kernel k-means, the initial centers are selected carefully by probability in a projected lower-dimensional subspace. The second contribution of this paper is a method speeding up the computation of the Gaussian kernel by projecting the data matrix into a lower-dimensional subspace. After the dimensionality of data is reduced, it effectively speeds up the computation of kernel k-means.

2. Related Work

In [9], random projection is adopted with iteratively increasing dimensionality for k-means clustering, which successfully obtains increasingly detailed resolutions and avoids local minima. Different methods for dimensionality reduction with feature selection and feature extraction for k-means clustering have been investigated by Boutsidis and Zouzias et al. in [10–12]. In [13], it is shown that dimensionality could be reduced by the use of PCA for data clustering with the objective function of k-means. An enhanced k-means method using a new method for centroid initialization with PCA is proposed by Napoleon and Pavalakodi in [14].

To improve clustering performance with kernels, in [15], an explicitly theoretical connection between kernel k-means and spectral clustering methods is presented by S. Dhillon et al. That connection leads to a weighted kernel k-means that monotonically decreases the normalized cut. In [16], a new clustering scheme is proposed by Zhang et al., which changes the clustering order from the sequence of samples to the sequence of kernels and employs a disk-based strategy to control the size of data. The new clustering scheme has been demonstrated to save ninety percent of the running time. A novel algorithm called approximate kernel k-means is proposed by Chitta et al. in [17], which tries to scale up the kernel-based clustering algorithm by using randomization to reduce both the computational complexity and the memory requirement.

However, with very large-scale datasets, memory requirements have always been a concern for kernel k-means. Incomplete Cholesky factorization is used to accelerate clustering and reduce the memory cost [18]. Its core idea is to utilize the product of a low-rank matrix and its transposition to approximate the complete kernel matrix. It is shown that with the decrease of the eigenvalues of the kernel matrix, the incomplete Cholesky factorization is exponentially convergent. Compared to k-means, kernel k-means can seize nonlinear structures with the cost of scaling poorly when the size of the matrix increases due to the fact that kernel methods always operate on the kernel matrix of the data. By employing approximately finite-dimensional feature maps based on spectral analysis, Amir Aradnia et al. [19] propose an approach to combine the advantages of both the linear and nonlinear methods. They suggest applying approximately finite-dimensional feature maps in kernel k-means to reduce the huge stored kernel matrix in the memory. Their explicit kernel Minkowski weighted k-means has been shown to be adaptive and capable of clustering in the new space. By adopting the Minkowski metric and feature weighting, the clustering performance improved and exhibited stronger robustness to noise features. Exploiting information from multiple views can result in better performance compared to clustering on a single view. For multiple-view clustering, a cluster-weighted kernel k-means method has been proposed [20]. It demonstrates high efficiency by assigning reasonable weights to related clusters among different views. The weight accounts for the significance of each cluster from each view to the final solution. In the meantime, it can also be learned automatically based on the intra-cluster similarity of the cluster different from all its corresponding clusters in different views.

In previous work, for improved centroid initialization, k-means++ mentioned in the introduction by Arthur and Vassilvitskii [8] is $O(log\,k)$-competitive with the optimal clustering. However, the question whether k-means++ yields an $O(1)$-approximation with probability $\frac{1}{poly(k)}$ or not is still open. In the work of Jaiswal and Garg [21], it is shown that the sampling algorithm gives an $O(1)$ approximation with probability $\Omega(\frac{1}{k})$ for any k-means problem instance where the data satisfies certain separation conditions. On the other hand, instances in [22] found that k-means++ achieves an approximation ratio of $(\frac{2}{3} - \epsilon)log\,k$. In [23] by Bhattacharya, Jaiswal, and Ailon, a simple two-dimensional dataset is presented, on which the seeding algorithm achieves an $O(log\,k)$ approximation ratio with probability exponentially small in k. In [24], the presented seeding algorithm works well even when more than one center is chosen in a single iteration with k-means++ parallelized, which drastically reduces the number of passes needed to obtain a good initialization for k-means. In [25], it has been shown that by mapping data points into randomized lower-dimensional subspace, the point-wise distances are preserved as well as conventional dimensionality reduction methods, such as principal component analysis (PCA).

Unlike the approaches in [18,20], our goal is to improve the computational efficiency for the kernel k-means and its initialization with k-means++ using random projection for dimensionality reduction such that all distances are computed in a much lower-dimensional space. We propose an efficient clustering method by (1) making k-means++ more efficient with random projection and (2) subsequently clustering data using kernel k-means in the projected lower-dimensional space. The intuition of the proposed method is to reduce the computational complexity by performing dimensionality reduction prior to both centroid initialization and kernel k-means. One of the advantages of the proposed method is that it is simple to code and analyze. The induced error is theoretically bound by the Johnson Lindenstrauss lemma (the JL lemma) (i.e., $\epsilon = \sqrt{\frac{20\ log\ n}{d}}$). Compared to the traditional dimensionality reduction approaches, such as feature selection, the features extraction we proposed would consider all the available features. This reduces the risk that relevant features might be mistakenly overlooked. Furthermore, in the efficient initialization we proposed, the user can strike a balance between accuracy and required learning time by the selection between kmFRP/kmIRP/kmBRP.

3. Our Method

3.1. Kernel k-Means with Random Projection

With kernel k-means, similarities are determined by replacing inner products with kernel functions. The complexity of kernel computation is $O(n^2D)$, which is a function of the dimensionality. The intuition of our method is to perform dimensionality reduction to reduce D with random projection prior to centroid initialization and kernel k-means clustering. The proposed method allows the computation of the kernel matrix and distances for initialization to be performed in a much lower-dimensional space in order to speed up the clustering process.

As the complexity of the kernel method with conventional kernel matrix computation $\mathbf{X} \cdot \mathbf{X^T}$ is $O(n^2D)$, the computation of the kernel matrix can be expensive when the dimensionality D is very large with genomic data, for example. We propose an approximation to the kernel matrix by using random projection, which maps the data points to a lower-dimensional space $\phi(\mathbf{X}) \cdot \phi(\mathbf{X})^T$, with the time complexity of $O(n^2)$, which is independent of D. With the same rationale as the method we propose for k-means++, combining random projection with kernel k-means allows the clustering process to be done in a lower-dimensional space without significant performance degradation.

With random projection and D-dimensional data vectors $\mathbf{x}$ with n data points, we have $\mathbf{X} \in \mathbb{R}^{n \times D}$ in which vectors are projected into much lower-dimensional subspace $\mathbb{R}^d$ with $d \ll D$, i.e.,

$$\mathbf{X}^{proj} = \mathbf{X} \cdot \mathbf{R} \tag{4}$$

where D denotes the original dimensionality, d denotes the reduced dimensionality, and $\mathbf{R}^{D \times d}$ is a random matrix with its entries as normal random variables $N(0,1) \cdot \frac{1}{\sqrt{d}}$. The complexity of this algorithm is of order $O(Ddn)$. If the data matrix $\mathbf{X}$ is very sparse with $\mathbf{s}$ non-zero entries per dimension, the complexity is of order $O(sdn)$, where s denotes the non-zero entries per dimension. $\mathbf{x} \in \mathbb{R}^D$ is projected into space $\mathbb{R}^d$, i.e., random projection is the mapping f: $\mathbb{R}^D \Rightarrow \mathbb{R}^d$ for all $\mathbf{u}, \mathbf{v}$ pairs in $\mathbb{R}^D$:

$$(1-\epsilon)\|\mathbf{u}-\mathbf{v}\|^2 \leq \|f(\mathbf{u})-f(\mathbf{v})\|^2 \leq (1+\epsilon)\|\mathbf{u}-\mathbf{v}\|^2 \tag{5}$$

where ϵ is the error bound that is determined as a function of the number of data points and dimensionality of the projected matrix, i.e., n and d as

$$\epsilon = \sqrt{\frac{20\ log\ n}{d}} \tag{6}$$

For kernel methods, in Lemma 1 of [1], there is a probability of at least $1-\delta$ that

$$\|X^TX - X^TR \cdot R^TX\|_2 \leq \epsilon_\delta \tag{7}$$

where the distance between the dot product in the original feature space and the dot product in the projected space is bounded by ϵ_δ. In our method with kernel k-means using random projection, the kernel matrix is approximated in a lower-dimensional space with the above error bound considered, shown in Algorithm 1.

Algorithm 1: Kernel k-means with random projection.

```
1  $\mathbf{R} \in \mathbb{R}^{D \times d}$ random matrix
2  $k(.,.)$denotes the kernel function.
3  $\mathbf{X}^{proj} = \mathbf{X} \cdot \mathbf{R}$
4  Choose initial centers $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ uniformly at random from $\mathbf{X}^{proj}$.
5  do
6      for $i \in 1, \ldots, n$ do
7          for $j \in 1, \ldots, k$ do
8              $dist_{i,j} = k(\mathbf{x}_i^{proj}, \mathbf{x}_i^{proj}) + k(\mathbf{c}_j^{proj}, \mathbf{c}_j^{proj}) - 2k(\mathbf{x}_i^{proj}, \mathbf{c}_j^{proj})$
9          end
10         Assign $\mathbf{x}_i^{proj}$ to $C_j$, when $min(dist_{i,j} \in dist_{i,1..k})$
11     end
12     for $i \in 1, \ldots, k$ do
13         $\mathbf{c}_i^{proj} = mean(C_i^{proj})$
14     end
15 while any of $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k$ is changed;
```

Since we propose to use random projection for both the center points initialization and clustering, the same projected matrix in a lower-dimensional space is used as input to both the k-means++ and kernel k-means. However, for kernel methods with random projection, the dot product is approximated like the approximated Euclidean distance using the JL lemma for the preservation of the point-wise Euclidean distance. In [2], an improved upper-bound on the error of the dot product with random projection is provided. Thus, to justify the upper bounded error of random projection for kernel k-means++ using both k-means++ and kernel k-means, the JL lemma and also the improved bound for kernel methods from [2] are considered.

In addition, in [26], it has been shown that initial values from k-means++ with the Euclidean distance could also improve the kernel k-means clustering performance measured by within-cluster-sum-of-squares (WCSS) and the time for convergence during training. In [27], it is also shown that the selection of k-means centroids in the instance space provides a good estimate of the kernel k-means centroids with the Gaussian kernel. It is computationally cheaper than the initialization performed in the kernel feature space.

3.2. *Efficient Initialization*

The k-means++ initialization algorithm is a popular initialization method for k-means with lower performance bounded by the statistical guarantee in [8]. We used random projection to approximate distances in the k-means++ algorithm to reduce the time complexity of $O(nkD)$. We let $Dist(\mathbf{x})$ denote the shortest distance from a data point $\mathbf{x}$ to the closest center already chosen. Since the preservation of point-wise distances in random projection is guaranteed by the JL lemma [28], the dimensionality of the input data can be reduced for $Dist(\mathbf{x})$ to remain close to its original value. The computation of the initialization can be reduced significantly, especially when the original dimensionality is very high.

An expensive step in the k-means++ algorithm is to compute the weights $Dist(\mathbf{x})^2$, which takes a new center $\mathbf{c}_i$ and chooses $\mathbf{x} \in \mathbf{X}$ with probability

$$\frac{Dist(\mathbf{x})^2}{\sum_{\mathbf{x} \in \mathbf{X}} Dist(\mathbf{x})^2} \tag{8}$$

Like kernel methods, after random projection, we do not need to have access to $\mathbf{x}$ directly as $Dist(\mathbf{x})^2$ is only required to be computed for choosing the initial center. We approximate $Dist(\mathbf{x})^2$ as the original dimensionality of $\mathbf{x}$, which can be very high and obtaining $Dist(\mathbf{x})^2$ can be expensive.

Unlike the work done by Boutsidis and Zouzias et al. [10–12], our idea is to find a good approximation to $Dist(\mathbf{x})^2$, which can be shown to be sufficiently close to the original value of $Dist(\mathbf{x})^2$ and at the same time, computationally simple. By approximating $Dist(\mathbf{x})^2$ in the k-means++ algorithm, we obtain an approximate algorithm to k-means++ with a complexity of $O(nk)$. The approximation to $Dist(\mathbf{x})^2$ can be obtained through random projections without perturbing the distances too much. We empirically show that these algorithms perform well on real-world datasets.

It has been shown that k-means++ is $O(log\ k)$-competitive with the optimal clustering. The algorithm has been shown to improve both the speed and the accuracy of Lloyd's algorithm for k-means. We take advantage of the performance guarantee of k-means++ provided by Theorem 3.1, which states that "if **C** is constructed with k-means++, then the corresponding potential function satisfies, $E[\phi]8(\ln k + 2)\phi_{OPT}$." [8].

Our proposed method for initialization is shown in Algorithm 3, which is called subroutine Algorithm 2. Algorithm 2 generates one of three types of random matrices for k-means++ with random projection, i.e., fixed random matrices, different random matrices for each iteration, and buffered random matrices (kmFRP, kmIRP, and kmBRP). Algorithm 3 obtains the random matrix generated in Algorithm 2, which computes the projected vectors with random projection and finds initial centers using data points projected into a lower-dimensional space in matrix $\mathbf{X}^{proj}$.

Algorithm 2: Proposed function to generate random matrix.

```
Input: type is the type of the random matrix, which can be kmFRP, kmIRP or
       kmBRP
       b is required for the buffer size if type is kmBRP
Output: $\mathbf{R} \in \mathbb{R}^{D \times d}$ random matrix
1  Create buffer B for the first time this subroutine is called such that
     $\mathbf{B} := \{\mathbf{R}_1, \mathbf{R}_2, \dots, \mathbf{R}_b\}$ with the loop:
2  for $i \in 1, \dots, b$ do
3      $\mathbf{R}_i = \overrightarrow{N}(0,1) \cdot \frac{1}{\sqrt{d}}$ 2
4  end
5  if type is kmIRP then
6      $\mathbf{R} = \overrightarrow{N}(0,1) \cdot \frac{1}{\sqrt{d}}$
7  else if type is kmBRP then
8      i = (random integer mod $b$) +1
9      $\mathbf{R} \leftarrow \mathbf{R}_i$
10 return R
```

$\overrightarrow{N}$(0,1) generates D × d normal distributed random numbers between 0 to 1.

3.2.1. k-Means++ with Fixed Random Projection (kmFRP)

kmFRP is the simplest variant of the initialization method. We approximate $\mathbf{X}$ by using $\mathbf{X}^{proj}$ such that $Dist(\mathbf{x})^2$ can be found with a time complexity of $O(d)$ instead of $O(D)$. $\mathbf{X}^{proj} \in \mathbb{R}^{n \times d}$ with projected vectors are computed before the seeding stage of k-means++ with random projections.

3.2.2. k-means++ with Iterative Random Projection (kmIRP)

With kmIRP, we also approximate $\mathbf{X}$ by using $\mathbf{X}^{proj}$. However, a new random projection is used at each iteration to produce $\mathbf{X}^{proj}$ with the aim of reducing the performance degradation measured in WCSS introduced by the approximation using random projection. This is like a stochastic gradient descent. With a different $\mathbf{X}^{proj}$ at each iteration, the approximation to $Dist(\mathbf{x})^2$ is different such that at the end of all iterations, $\mathbf{X}^{proj}$ better approximates the original high-dimensional vectors $\mathbf{x}$ with averaging. However, the disadvantage is that the computation for random projection is done at each iteration, so it is computationally more expensive.

3.2.3. k-Means++ with Buffered Random Projection (kmBRP)

Buffering a fixed number of random matrices is also considered for the initial center selection. The aim is to combine the advantages of the previous two variants with a limited number of random projections for efficiency. Although multiple random projections could help to reduce the bias in approximation, a new random matrix at each iteration may not be necessary. At the initialization stage, we have a buffer $\mathbf{B}$ with b random matrices that, $\mathbf{B} := \{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_b\}$. At each iteration, a random index i is generated for $\mathbf{R_i} \in \mathbf{B}$.

k-means++ with fixed random projection (kmFRP) is the simplest and the most computationally efficient variant of our method, but it may lead to a larger degradation of performance, as discussed previously. k-means++ with iterative random projection (kmIRP) theoretically provides the least biased clustering at different iterations with more computation for random projections. k-means++ with buffered random projection (kmBRP) is something in the middle between kmFRP and kmIRP that strikes a balance for good optimization performance while maintaining reasonable computational efficiency. With the free parameter b, which is the buffer size, the user can control the trade-off between optimization performance and computational efficiency.

Algorithm 3: Proposed k-means++ approximation with different types of random projection.

```
Input: type is the type of the random matrix, which can be kmFRP, kmIRP
       or kmBRP
Input: b is required for the buffer size if type is kmBRP
Output: k initial centers selected
1 Choose initial center $\mathbf{c}_1$ uniformly at random from $\mathbf{X}$
2 $\mathbf{R}_i = \overrightarrow{N}(0,1) \cdot \frac{1}{\sqrt{d}}$
3 for $i \in 2, \ldots, k$ do
4     $\mathbf{X}^{proj} = \mathbf{X} \cdot \mathbf{R}$
5     Choose $\mathbf{c}_i = \mathbf{x} \in \mathbf{X}^{proj}$ with probability $\frac{Dist(\mathbf{x})^2}{\sum_{\mathbf{x} \in \mathbf{X}^{proj}} Dist(\mathbf{x})^2}$
6     if type is either kmIRP or kmBRP then
7         R = call Algorithm 2 with type, b
8 end
```

3.3. Complexity

Random projection is known to have a time complexity of $O(sdn)$ instead of $O(Ddn)$ with sparse matrices, where s denotes non-zero entries per dimension. Sparse data are common in various applications, including text classification and bioinformatics data (e.g., gene expression data). k-means++ has the complexity of $O(Dnk)$. With random projections, the complexity of k-means++ becomes of order $O(dnk)$ as we can avoid computing for all D dimensions. In practice, with our JL lemma application in the centers' initialization in k-means++, dimensionality can be reduced to a few hundred, e.g., 200. The lemma requires no assumption on the original data [28]. Therefore, d in $O(dnk)$ can be replaced by a constant. With the constant, $O(dnk)$ becomes $O(nk)$.

4. Experiments

4.1. Datasets

We used six datasets to evaluate the improvement of our method. The first two are computer vision datasets, namely MNIST [29] and ImageNet [30]. Another two datasets are from bioinformatics datasets, called SMK-187 [31] and GLI-85 [32] (the SMK-187 and GLI-85 datasets could be downloaded from https://jundongl.github.io/scikit-feature/OLD/datasets_old.html accessed date 27 July 2021). The last two datasets are for text mining, called 20 newsgroups [33] and Reuters-21578 [34].

MNIST is a ten-class database for handwritten digit recognition. There are 60,000 samples and 10,000 samples in the training set and the testing set, respectively. In the testing set, there are 1000 samples per class label. Every sample consists of 28 by 28 pixels (i.e., 784 features) with which the value is inside the interval [0,255], and represents the greyscale of that pixel. We evaluate clustering performance by comparing how well different methods optimize clustering with the WCSS measure. In our experiments, we only sample a subset of 5000 data points from the training set. Thus, the dataset to be examined consists of 5000 data points with dimensionality 784 (i.e., n = 5000, D = 784).

The ImageNet [30] database consists of over 14.2 million images that are organized according to the WordNet hierarchy. Each node in this hierarchy represents a concept (known as the "synset"), and all the images are indexed by over 21-thousand synsets. In our analysis, we chose 14,112 images from ten cat-like synsets (namely, an Egyptian cat, a foumart, a lynx, a Madagascar cat, a mountain lion, a Persian cat, a Siamese cat, a skunk, a tabby cat, and a tiger cat). They are all from the leaf nodes in the hierarchy. We called this dataset the ImageNet-Cat dataset. Figure 1 shows an example from each synset used to form this dataset. We used the bag-of-visual-words features of these images from image-net.org. It consists of 1000 visual-words features per photo. In our experiments, we only sampled 2000 images from the ImageNet-Cat dataset for evaluation with $n = 2000$ and $D = 1000$.

Figure 1. Ten different types of cats from ImageNet.

SMK-187 is a dataset with gene expression data from smokers with lung cancer and also samples from smokers without lung cancer. There are 19,993 features in each sample in SMK-187. This is a two-class dataset with $n = 187$ and $D = 19,993$.

GLI-85 is a dataset with transcriptional profiling of gliomas. The dataset is collected from 74 patients where transcriptional profiling has been applied to 85 gliomas. The 74 patients studied were analyzed if their initial tumor was diagnosed as a Grade III or IV glioma of any histologic type on initial surgical treatment. There are 22,283 features in each sample in GLI-85 with $n = 85$ and $D = 22,283$.

The 20-Newsgroups dataset contains a collection of newsgroup articles from 20 different newsgroups. It consists of approximately 1000 documents from each of these newsgroups. In our experiment, we only clustered data from 5 of these newsgroups, including alt.atheism, os.ms-windows, sport.baseball, sci.space and religion.misc. With the corpus, we applied a number of data pre-processing steps that include converting characters to lower cases, removing punctuations, removing white spaces, and removing stop words in English. Stop words are 174 predefined English common words that are believed to contain very little information for text mining (such as "we", "your", "the", "any", and "only"). Terms in the corpus are columns, while documents are the rows. Columns with a sparsity of 98% or above were removed, and only 2000 data points were sampled before each iteration of the evaluation. This resulted in a subsampled set with 2000 data points and 929 dimensions, i.e., $n = 2000$, $D = 929$.

Reuters-21578 contains a collection of 21,578 news articles as Reuters newswires in 1987. Documents were assembled and indexed with categories by staff at Reuters Ltd. and Carnegie Group, Inc. The Reuters-21578 dataset that we used is the well-formed dataset assembled in the text mining package (i.e., tm) in R [35]. In our experiment, we only selected news from the *acq* and *earn* topic categories with 2362 and 3945 news articles, respectively. We removed the terms with sparsity 99.5% or above before clustering. Thus, there are 6307 data points with 974 dimensions ($n = 6307$, $D = 974$) in the document-term matrix.

4.2. Environments

Our experiments (the source code of the project would be provided upon publishing of the paper through: https://github.com/janchanyk/kmeans-plusplus-RP, accessed date 27 July 2021) were run on a 64-bit platform with an Intel 3.1 GHz Quad-Core CPU, equipped with 32 GB of memory. All the experiments were run on the same platform.

4.3. Performance Metrics

We evaluated the results by observing the performance and the execution time of each of the experiments. The way we evaluated the performance was just to use the standardly potential function of the k-means problem, i.e., the within-cluster sum of squares (WCSS). However, we used it to measure the distance of each data point to its closest initial center after the initialization. We also used this metric to evaluate the overall clustering performance. Our objective is to test how well the proposed algorithms minimize the potential when compared to the baseline. The potential, WCSS, could be defined as

$$\sum_{i \in k} \sum_{\mathbf{x} \in \mathbf{C}_i} \|\mathbf{x} - \mathbf{c}_i\|^2 \tag{9}$$

where $\mathbf{C}_i$ is the i-th cluster, and $\mathbf{c}_i$ is the initial center point of the i-th cluster.

Since we used dimensionality reduction before our proposed initialization algorithm and clustering, the clustering performance in that lower-dimensional space should not be compared to that of the original input space. Therefore, we obtained the cluster label of each point and computed WCSS of these points to the center of mass of each cluster, back in the original input space.

Another evaluation metric is the execution time. For the experiments of our proposed algorithm on k-means++, we only counted for the time used in the initial center selection. Our objective is to see the time used by the algorithms for the center points initialization, instead of the time consumed by k-means. In the experiments on clustering, we also used this time metric to see the time reduced in our proposed method. The execution time of clustering includes the processing time of both random projection and kernel k-means clustering.

4.4. Experiments on the Efficient Centroid Initialization

On the two bioinformatics datasets, namely GLI-85 and SMK-187, we first established the baseline by running k-means++ in the original input space. Subsequently, we ran the efficient centroid initialization method (i.e., kmFRP). We iterated each of the above experiments ten times to measure its average performance (in WCSS). We compared the WCSS from our proposed method against the baseline.

4.5. Experiments on the Kernel k-Means Clustering

On each dataset, we first established the baseline by running kernel k-means and kernel k-means++ in the original input space. Subsequently, the data were projected into a lower-dimensional feature space, and kernel k-means and kernel k-means++ are evaluated in that projected space. The Gaussian kernel was adopted when the data were fitted to the kernel k-means clustering. For the experiments in the projected space, we increased the number of dimensions from 2 to 500 to see its impact on the clustering performance. We iterated each experiment ten times to measure its average performance and execution time.

4.6. Findings

4.6.1. Performance of Our Method for Centroid Initialization

We first evaluated the initialization performance when the k-means++ was run in the projected subspace with our method. We compared the average performance in WCSS of our proposed k-means++ approximation method, with the average performance of the original version of the k-means++.

Both Figures 2 and 3 show that the initialization performance in WCSS by our method is close to the baseline. The WCSS by our k-means++ approximation is always within the error bar (i.e., one standard deviation) of the WCSS distribution by the original k-means++. When the *k* is increasing, the error bar of the WCSS distribution by the original k-means++ is narrowing down, and the performance in WCSS by our k-means++ approximation method is still close to the baseline and falls within the error bar.

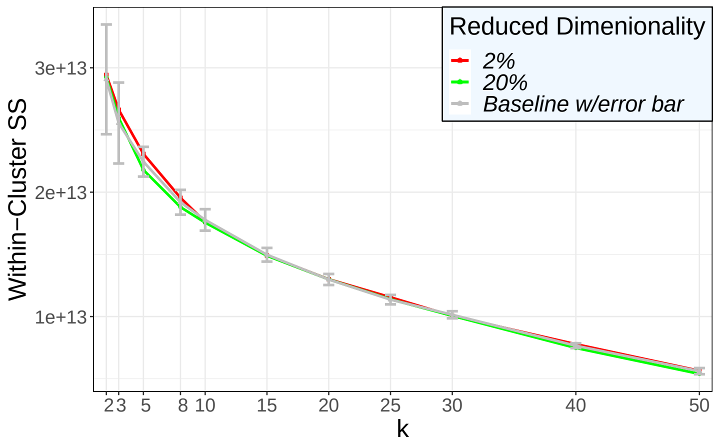

Figure 2. WCSS of initial centers with the GLI-85 Dataset.

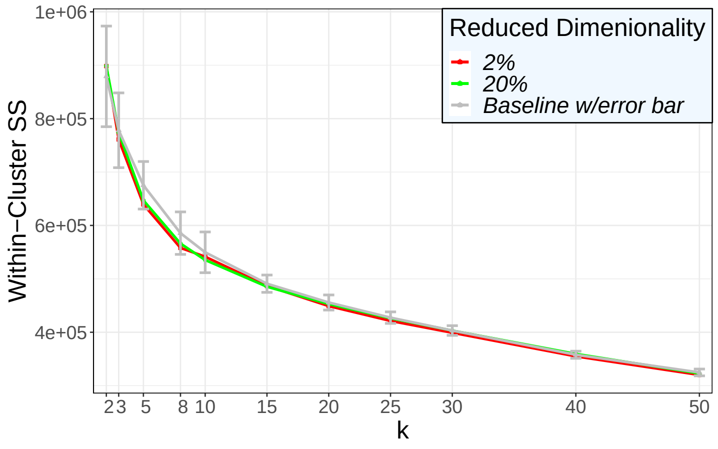

Figure 3. WCSS of initial centers with the SMK-187 Dataset.

We then evaluated the difference in initialization performance from our method against the baseline. We evaluated the degradation of the average performance of our proposed k-means++ approximation algorithms by comparing them with the original version of k-means++. The degradation of the average performance ϵ in our experiment is defined as,

$$\frac{|WCSS_{rp} - WCSS_{orig}|}{WCSS_{orig}} \tag{10}$$

where $WCSS_{rp}$ is the distance from our method after random projection and $WCSS_{orig}$ is the distance from the original version of k-means++.

The values ϵ and the execution time are compared for different methods in Tables 1 and 2, respectively, to show the seeding performance on GLI-85, SMK-187 and the Reuters-21578 dataset.

Table 1. Comparison with ϵ.

Dataset	d	k = 8	k = 10	k = 20	k = 50
Reuters-21578 (D = 2003)	200	2.08%	1.81%	0.60%	0.42%
	300	1.85%	0.86%	1.37%	0.47%
	401	0.12%	0.40%	0.53%	0.02%
GLI-85 (D = 22283)	2228	0.5%	0.73%	1.06%	0.19%
	3342	1.53%	2.59%	0.93%	0.81%
	4457	2.03%	1.10%	0.20%	3.94%
SMK-187 (D = 19993)	1999	1.73%	0.40%	0.16%	1.40%
	2999	3.74%	1.98%	0.02%	0.01%
	3999	3.33%	2.62%	0.62%	0.58%

After dimensionality reduction by using random projection, there is no significant change in WCSS. The larger the k, the smaller the ϵ value.

Table 2. Training time comparison.

Dataset	d	k = 8	k = 10	k = 20	k = 50
Reuters-21578 (D = 2003)	baseline	20.48	33.47	141.22	920.74
	200	8.61	11.49	26.28	135.16
	300	10.65	14.39	32.59	132.22
	401	13.03	17.89	41.63	175.59
GLI-85 (D = 22283)	baseline	3.77	5.93	22.44	142.02
	2228	4.77	5.33	8.68	33.63
	3342	6.7	7.24	11.21	37.07
	4457	8.85	9.30	13.34	40.89
SMK-187 (D = 19993)	baseline	5.96	8.88	32.27	194.97
	1999	5.92	6.35	9.86	31.76
	2999	8.48	8.97	12.97	38.41
	3999	10.99	11.56	16.07	44.94

The larger the dataset, the bigger the time improvement. There is a computational advantage for the proposed method with reduced execution time when k $\geq$ 20 in general.

Throughout the experiments on the bioinformatics datasets of GLI-85, SMK-187, and the Reuters-21578 dataset, it is found that the dimensionality reduction using random projection does not induce any significant increase to the ϵ of the center point initialization. The figures also show that the dimensionality could be reduced down to 10% of the original dimensionality, and the center point initialization still performs well with a small ϵ. In general, it is also found that the larger the number of clusters (i.e., the k), the smaller the ϵ induced by our approximation.

In the experiment on the Reuters-21578 dataset, since it is the largest dataset among three datasets, (i.e., $n = 21,578, D = 2003$), time improvement on the center point initialization exists in most of the experiments with different parameters. For the GLI-85 and SMK-187 datasets, the major time reduction only happens on the large k. When the k is small, the cost of random projection offsets the cost reduced from the center point initialization. In other words, the proposed solution is especially fit for solving problems in high dimensionality and when the k is large.

4.6.2. Clustering Performance of Our Method

We evaluated the clustering performance on six different real-world datasets. Figure 4 indicates that the overall execution time improvement in the projected space can be seen in all experiments. With enough dimensionality such that no performance degradation is observed, it is shown that our method generally reduces the execution time by over 50%. With the bioinformatics dataset, kernel k-means++ in the projected space runs 10–20 times faster than the ordinary kernel k-means in the original input data space. Table 3 summarizes the time reduced in our experiments. With the execution time of the baseline, it is not always the case that k-means++ can shorten the clustering process. The execution time on the bioinformatics dataset GLI-85 and SMK-187 indicate that, on a dataset with high dimensionality and limited data points (e.g., bioinformatics data), the k-means++ takes much longer time to find the initial centers than the computation of kernel k-means clustering.

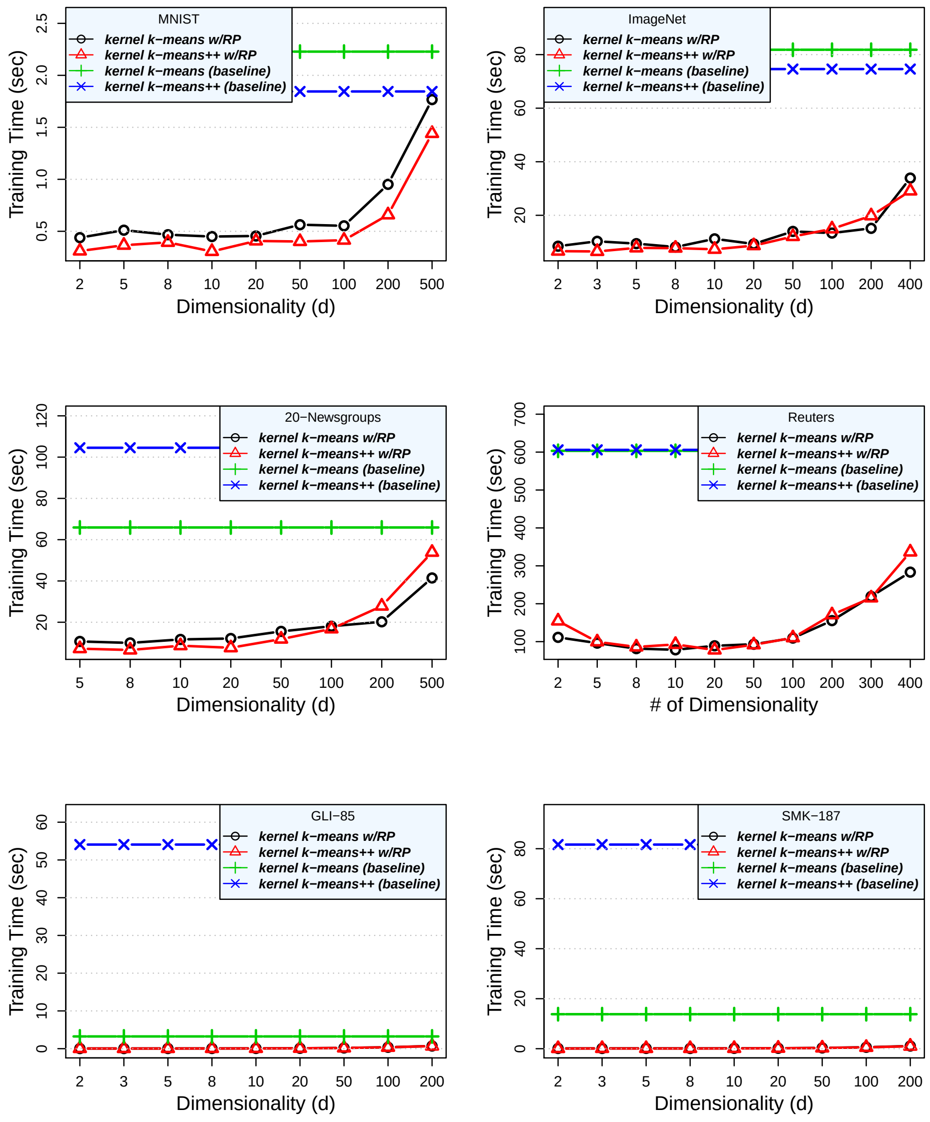

Figure 4. Reduced training time.

Table 3. Training time reduced with our method compared to that of vanilla kernel k-means. In general, the higher the original dimensionality (D), the more the speed-up.

Dataset	**D**	**d**	*Proposed* [1]	*Baseline* [2]	**Speed-up**
MNist	784	100	0.42	2.22	5.4×
ImageNet-Cat	1000	200	19.80	81.82	4.13×
SMK-187	19,993	100	0.53	13.80	26.1×
GLI-85	22,283	100	0.36	3.24	9.10×
Reuters	929	100	110.2	603.3	5.47×
20Newsgroups	947	200	27.8	65.9	2.37×

[1] The execution time of our proposed method. [2] The execution of the vanilla kernel k-means.

Figure 5 illustrates the clustering performance of the proposed method against the increasing dimensionality. There are two horizontal lines in green and blue, corresponding to kernel k-means and kernel k-means++, respectively, in the input feature space (the baseline) with the clustering performance of our methods in black and red against an increasing number of dimensions.

The results with the six experiments generally indicate that the clustering performance of the kernel k-means++ is either better than or close to that of kernel k-means in the original input space or the projected feature space. In the projected data space, WCSS is much higher than that of the baseline only when the dimensionality is very low. In other words, the performance deteriorates a lot when the data have been projected into an extremely low dimensional space. The clustering performance converges to the performance of the baseline when the number of dimensions increases. In general, the performance with projection converges to that of the baseline when the dimensionality is between 100 and 200.

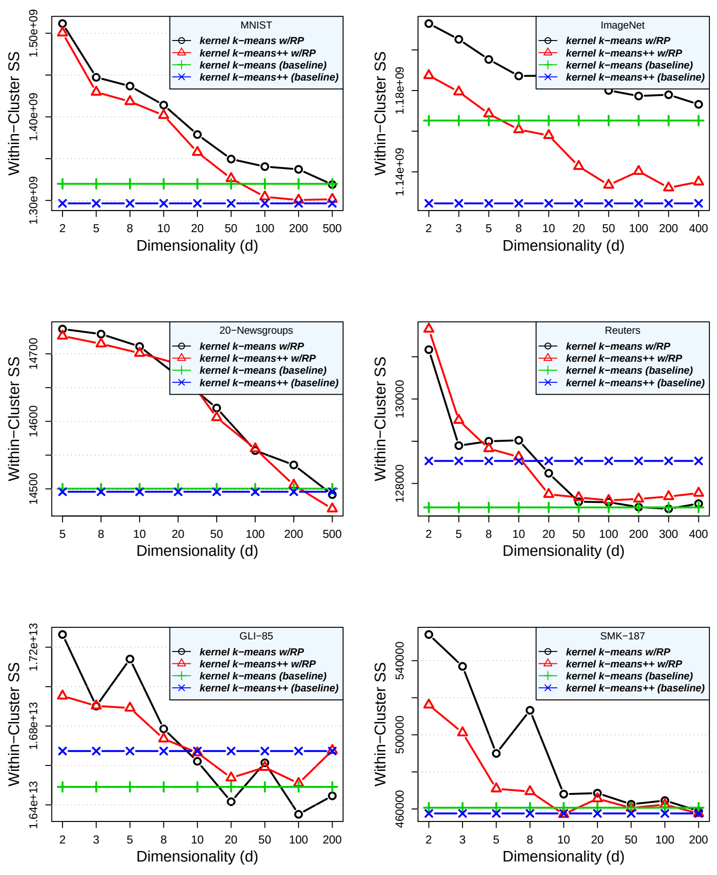

Figure 5. WCSS of approximation close to original kernel k-means.

5. Discussions

5.1. Optimal Dimensionality of the Projected Space

We evaluated our methods on six datasets, and the performance deteriorates a lot when the dimensionality is extremely low. The performance converges to the result of the baseline until the optimal dimensionality is reached. The optimal dimensionality varies among different datasets. In the experiment on the ImageNet-Cat dataset, if we pick synsets very different from each other (e.g., newspaper vs. tiger cat), which means the data instances among these synsets, with bag-of-visual-words scale-invariant feature transform (SIFT) features, are very separable. In such cases, this problem could be solved even in a very low-dimensional (i.e., even below 10) feature space. In other words, we may start the algorithm in a low-dimensional space and observe the clustering performance improvements by increasing the dimensionality until it saturates.

5.2. Ablation Study on Our Method

The performance of our proposed method, kernel k-means++ with random projection shown with the red lines in Figure 5 is close to the performance of kernel k-means in the original input space in green lines. For the training time in Figure 4, we can observe the time consumption of different components of the method. For kernel k-means in the projected feature space, the black lines in the figure correspond to experiments without k-means++ initialization and red lines correspond to experiments with k-means++. It can be observed that both kernel k-means and kernel k-means++ take more time to complete without random projection. The execution time increases linearly when the projected dimensionality increases. From the clustering performance point of view, in all our results, kernel k-means++ performs close to kernel k-means whether or not clustering is performed in the original input space or the projected low-dimensional space.

5.3. Reliability, Stability and Robustness

Table 4 presents some other clustering performance measurements that help assess the reliability, stability and robustness of the proposed method.

Table 4. Other clustering accuracy measurements of our experiments.

Dataset	d [1]	% Changes in WCSS [2]	Stdev / Mean of Kernel Kmeans++ w/RP [3]	Stdev / Mean of Vanilla Kernel Kmeans [4]
MNIST	100	−1.18%	1.83%	1.6%
	200	−1.47%	1.81%	
	500	−1.40%	1.70%	
ImageNet	100	−2.14%	4.78%	4.32%
	200	−2.84%	5.01%	
	500	−2.59%	3.99%	
20 Newsgroup	100	0.41%	0.53%	0.5%
	200	0.04%	0.69%	
	500	−0.21%	0.75%	
Reuters	200	0.16%	0.40%	0.23%
	300	0.20%	0.38%	
	400	0.27%	0.81%	
GLI-85	50	0.60%	1.55%	1.41%
	100	0.11%	1.53%	
	200	1.12%	1.48%	
SMK-187	50	−0.02%	2.13%	1.35%
	100	0.37%	1.96%	
	200	−0.67%	2.16%	

[1] The projected dimensionality d. [2] Percentage change in result WCSS by the proposed kernel k-means++ with random projection, compared to the vanilla kernel k-means. [3] Fluctuation of the result WCSS by the proposed kernel k-means++ with random projection (kernel k-means++ w/RP). [4] Fluctuation of the result WCSS by the vanilla kernel k-means.

5.3.1. Reliability

Since our method is an approximation to the kernel k-means, we assessed the accuracy of our approximation by observing the percentage change in WCSS compared to the vanilla kernel k-means. The percentage changes in WCSS of Table 4 show that in all the six datasets, the resulting WCSS from our method is very close to that (i.e., below one percent) of the vanilla kernel k-means. Some cases record negative changes in WCSS, which represents that the clustering performance from our method is even better than the vanilla kernel k-means.

5.3.2. Robustness

For algorithm robustness, it usually represents the comparison between the training error and the test error for supervised learning. In our case of clustering, we are assessing the robustness by considering the noise associated with the random projection. In our experiments, when the projected dimensionality is small, the noise induced by the random projection is theoretically higher than that of higher projected dimensionality. In our experiments with six datasets, when $d > 200$, percentage changes in WCSS are less than 1% in most of the cases. In other words, the proposed method remains robust when the projected dimensionality d is greater than 200.

5.3.3. Stability

In our experiments, for each combination of the dataset and d, we iterated the evaluation of our proposed method ten times. We can estimate the stability by means of fluctuation (i.e., stdev/means) of the resulting WCSS from our proposed method. It is found that fluctuation of the WCSS is only below 5% of the mean WCSS, and most of the cases are below 2%. They are close to the fluctuation of the resulting WCSS from vanilla kernel k-means. On the other hand, it is also found that the fluctuation of WCSS from our approximation approach is always higher (by less than one percent) than that from the vanilla kernel k-means. In other words, the speed-up from our proposed solution comes with a cost, but this cost is relatively small in our experiment with six datasets.

6. Conclusions

We propose a method to utilize dimensionality reduction before k-means++ centroid initialization and kernel k-means clustering. To justify the clustering performance of the proposed method using random projection for both k-means++ and kernel k-means, the JL lemma and the improved bound for kernel methods from [2] are considered. Experimental results indicate that the performance with our approach is close to that of kernel k-means with k-means++ initialization without significant degradation. On a high dimensional dataset in our experiment (n = 187, D = 19,993), it is shown that the speed-up of the proposed algorithms is 26 times compared to that of vanilla kernel k-means. The result demonstrates that the proposed method speeds up the kernel k-means, by means of dimensionality reduction using random projection, without significant performance degradation.

6.1. Managerial and Academic Implications

Clustering is one of the major approaches in unsupervised learning. They are heavily used in many areas like spam filtering, marketing and document analysis, etc., and k-means is one of the most popular methods in clustering. However, k-means does not scale very well when the dimensionality of a dataset is very high. The method we proposed tackles this weakness and improves the scalability of k-means++ and kernel k-means.

6.2. Limitations

The method we propose only reduces the dimensionality from D to d, and the complexity of the method is reduced from $O(nkD)$ to $O(nkd)$. On the other hand, the number of data points (i.e., n) remains unchanged. Therefore, the execution time may remain sensitive to n.

6.3. Future Studies and Recommendations

In the proposed method, we reduce the computational cost of k-means++ and kernel k-means using random projection. The random projection could be further sped up by some advanced approaches [36–39]. It could be interesting to consider these approaches to further speed up the k-means clustering algorithm.

Author Contributions: Conceptualization, J.Y.K.C. and A.P.L.; methodology, J.Y.K.C. and A.P.L.; software, J.Y.K.C.; validation, J.Y.K.C. and A.P.L.; formal analysis, J.Y.K.C. and A.P.L.; resources, J.Y.K.C.; data curation, J.Y.K.C.; writing—original draft preparation, J.Y.K.C. and A.P.L.; writing—review and editing, J.Y.K.C., A.P.L. and Y.X.; visualization, J.Y.K.C.; supervision, A.P.L.; project administration, A.P.L.; funding acquisition, A.P.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Macau Science and Technology Development Fund grant number 0025/2019/AKP.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in publicly accessible repository.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Paul, S.; Boutsidis, C.; Magdon-Ismail, M.; Drineas, P. Random Projections for Support Vector Machines. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*; Proceedings of Machine Learning Research; Carvalho, C.M., Ravikumar, P., Eds.; PMLR: Scottsdale, AZ, USA, 2013; Volume 31, pp. 498–506.
2. Kabán, A. Improved Bounds on the Dot Product under Random Projection and Random Sign Projection. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 August 2015; Cao, L., Zhang, C., Joachims, T., Webb, G.I., Margineantu, D.D., Williams, G., Eds.; ACM: New York, NY, USA, 2015; pp. 487–496. [CrossRef]
3. Keivani, O.; Sinha, K. Random projection-based auxiliary information can improve tree-based nearest neighbor search. *Inf. Sci.* **2021**, *546*, 526–542. [CrossRef]
4. Liberti, L.; Poirion, P.; Vu, K.K. Random projections for conic programs. *arXiv* **2021**, arXiv:2101.04182.
5. Heusinger, M.; Schleif, F. Random Projection in supervised non-stationary environments. In Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2020, Bruges, Belgium, 2–4 October 2020; pp. 405–410.
6. Cao, T.; Vo, C.; Nguyen, S.; Inoue, A.; Zhou, D. A Kernel k-Means-Based Method and Attribute Selections for Diabetes Diagnosis. *J. Adv. Comput. Intell. Intell. Inform.* **2020**, *24*, 73–82. [CrossRef]
7. Paul, D.; Chakraborty, S.; Das, S.; Xu, J. Kernel k-Means, By All Means: Algorithms and Strong Consistency. *arXiv* **2020**, arXiv:2011.06461.
8. Arthur, D.; Vassilvitskii, S. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*; SODA '07; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2007; pp. 1027–1035.
9. Cardoso, Â.; Wichert, A. Iterative random projections for high-dimensional data clustering. *Pattern Recognit. Lett.* **2012**, *33*, 1749–1755. [CrossRef]
10. Boutsidis, C.; Zouzias, A.; Mahoney, M.W.; Drineas, P. Stochastic Dimensionality Reduction for K-means Clustering. *arXiv* **2011**, arXiv:1110.2897.
11. Boutsidis, C.; Zouzias, A.; Drineas, P. Random Projections for k-means Clustering. *arXiv* **2010**, arXiv:1011.4632v1.
12. Boutsidis, C.; Zouzias, A.; Mahoney, M.W.; Drineas, P. Randomized Dimensionality Reduction for k-Means Clustering. *IEEE Trans. Inform. Theory* **2015**, *61*, 1045–1062. [CrossRef]
13. Ding, C.; He, X. K-means Clustering via Principal Component Analysis. In *Proceedings of the Twenty-first International Conference on Machine Learning*; ICML '04; ACM: New York, NY, USA, 2004; p. 29. [CrossRef]
14. Napoleon, D.; Pavalakodi, S. A New Method for Dimensionality Reduction Using KMeans Clustering Algorithm for High Dimensional Data Set. *Int. J. Comput. Appl.* **2011**, *13*, 41–46. [CrossRef]
15. Dhillon, I.S.; Guan, Y.; Kulis, B. Kernel K-Means: Spectral Clustering and Normalized Cuts. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–28 July 2010; KDD '04; Association for Computing Machinery: New York, NY, USA, 2004; pp. 551–556. [CrossRef]

16. Zhang, R.; Rudnicky, A.I. A large scale clustering scheme for kernel K-Means. In Proceedings of the 2002 International Conference on Pattern Recognition, Quebec City, QC, Canada, 11–15 August 2002; Object Recognition Supported by User Interaction for Service Robots; Volume 4, pp. 289–292.
17. Chitta, R.; Jin, R.; Havens, T.C.; Jain, A.K. Approximate Kernel K-Means: Solution to Large Scale Kernel Clustering. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011; KDD '11; Association for Computing Machinery: New York, NY, USA, 2011; pp. 895–903. [CrossRef]
18. Chen, L.; Zhou, S.; Ma, J. Fast Kernel k-means Clustering Using Incomplete Cholesky Factorization. *arXiv* **2020**, arXiv:2002.02846.
19. Aradnia, A.; Haeri, M.A.; Ebadzadeh, M.M. Adaptive Explicit Kernel Minkowski Weighted K-means. *arXiv* **2020**, arXiv:2012.02805.
20. Liu, J.; Cao, F.; Gao, X.; Yu, L.; Liang, J. A Cluster-Weighted Kernel K-Means Method for Multi-View Clustering. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020*; AAAI Press: Palo Alto, CA, USA, 2020; pp. 4860–4867.
21. Jaiswal, R.; Garg, N. Analysis of k-Means++ for Separable Data. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques. In Proceedings of the 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, 15–17 August 2012; pp. 591–602. [CrossRef]
22. Brunsch, T.; Röglin, H. A Bad Instance for K-means++. *Theor. Comput. Sci.* **2013**, *505*, 19–26. [CrossRef]
23. Bhattacharya, A.; Jaiswal, R.; Ailon, N. A Tight Lower Bound Instance for k-means++ in Constant Dimension. In *Theory and Applications of Models of Computation, Proceedings of the 11th Annual Conference, TAMC 2014, Chennai, India, 11–13 April 2014*; Springer International Publishing: Cham, Switherland, 2014; pp. 7–22. [CrossRef]
24. Bahmani, B.; Moseley, B.; Vattani, A.; Kumar, R.; Vassilvitskii, S. Scalable K-means++. *Proc. VLDB Endow.* **2012**, *5*, 622–633. [CrossRef]
25. Bingham, E.; Mannila, H. Random Projection in Dimensionality Reduction: Applications to Image and Text Data. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26–29 August 2001; KDD '01; pp. 245–250. [CrossRef]
26. Papp, D.; Szűcs, G. MMKK++ algorithm for clustering heterogeneous images into an unknown number of clusters. *ELCVIA Electron. Lett. Comput. Vis. Image Anal.* **2018**, *16*, 30. [CrossRef]
27. Oglic, D.; Gärtner, T. Nyström Method with Kernel K-Means++ Samples as Landmarks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 2652–2660.
28. Johnson, W.; Lindenstrauss, J. Extensions of Lipschitz mappings into a Hilbert space. *Contemp. Math.* **1984**, *26*, 189–206.
29. LeCun, Y.; Cortes, C. MNIST Handwritten Digit Database 2010. Available online: http://yann.lecun.com/exdb/mnist/ (accessed on 27 July 2021).
30. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
31. Spira, A.; Beane, J.E.; Shah, V.; Steiling, K.; Liu, G.; Schembri, F.; Gilman, S.; Dumas, Y.M.; Calner, P.; Sebastiani, P.; et al. Airway epithelial gene expression in the diagnostic evaluation of smokers with suspect lung cancer. *Nat. Med.* **2007**, *13*, 361–366. [CrossRef] [PubMed]
32. Freije, W.A.; Castro-Vargas, F.E.; Fang, Z.; Horvath, S.; Cloughesy, T.; Liau, L.M.; Mischel, P.S.; Nelson, S.F. Gene Expression Profiling of Gliomas Strongly Predicts Survival. *Cancer Res.* **2004**, *64*, 6503–6510. [CrossRef] [PubMed]
33. 20 Newsgroups Dataset. Available online: https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups (accessed on 27 July 2021).
34. Reuters-21578 Dataset. Available online: https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection (accessed on 27 July 2021).
35. Ingo, F.; Kurt, H.; David, M. Text Mining Infrastructure in R. *J. Stat. Softw.* **2008**, *25*. [CrossRef]
36. Liberty, E.; Ailon, N.; Singer, A. Dense Fast Random Projections and Lean Walsh Transforms. In Proceedings of the Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX 2008, and 12th International Workshop, RANDOM 2008, Boston, MA, USA, 25–27 August 2008; pp. 512–522. [CrossRef]
37. Li, P.; Hastie, T.; Church, K. Very sparse random projections. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, 20–23 August 2006; Volume 2006, pp. 287–296. [CrossRef]
38. Ailon, N.; Chazelle, B. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM J. Comput.* **2009**, *39*, 302–322. [CrossRef]
39. Kane, D.M.; Nelson, J. Sparser Johnson-Lindenstrauss Transforms. *arXiv* **2014**, arXiv:1012.1577.