Approximate Nearest Centroid Embedding for Kernel k-Means

Ahmed Elgohary, Ahmed K. Farahat, Mohamed S. Kamel, and Fakhri Karray
University of Waterloo, Waterloo, Canada N2L 3G1
{aelgohary,afarahat,mkamel,karray}@uwaterloo.ca

Abstract

This paper proposes an efficient embedding method for scaling kernel k-means on cloud infrastructures. The embedding method allows for approximating the computation of the nearest centroid to each data instance and, accordingly, it eliminates the quadratic space and time complexities of the cluster assignment step in the kernel k-means algorithm. We show that the proposed embedding method is effective under memory and computing power constraints, and that it achieves better clustering performance compared to other approximations of the kernel k-means algorithm.

1 Introduction

Modern cloud computing infrastructures tend to be composed of several commodity nodes, each of which is of a very limited memory and computing power [2, 6, 11]. The kernel k-means is a commonly-used data clustering algorithm that is capable of handling datasets of arbitrary complex structures and capturing nonlinearity in the feature space. However, due to its kernelized nature, it is always challenging to parallelize the kernel k-means algorithm [7] on cloud infrastructures to handle massively distributed datasets. In this paper, we present an approximate kernel k-means algorithm that better suits modern cloud infrastructures. Our approach is based on computing a low-dimensional embedding for each data instance by the proposed Approximate Nearest Centroid (APNC) embedding method. The resulting embeddings are then used to approximate the cluster assignment step obviating the need to work with kernels in kernel k-means iterations. We start in Section 2 by giving a necessary background on the kernel k-means followed by a review of recently proposed approximations. We describe our embedding method in Section 3 and demonstrate the effectiveness of the proposed approach through empirical evaluation in Section 4.

2 Kernel k-Means and Approximations

The kernel k-means [7] is a variant of the well-known k-means algorithm [10] which works on a kernel matrix that encodes all pairwise inner-products between n data instances mapped to a typically high-dimensional space. Let $\boldsymbol{\phi}^{(i)}$ denote the mapping of a data instance i into a kernel space endowed implicitly by a kernel function $\kappa(.,.)$. The kernel k-means algorithm tries to assign each data instance i to one of k possible clusters such that the ℓ_2-loss function $\sum_{c=1}^{k}\sum_{i\in\Pi_c}||\boldsymbol{\phi}^{(i)}-\bar{\boldsymbol{\phi}}^{(c)}||_2$ is minimized [7]. We use Π_c to denote the set of data instances assigned to the cluster c and $\bar{\boldsymbol{\phi}}^{(c)}$ to denote the centroid of the data instances in Π_c (i.e. $\bar{\boldsymbol{\phi}}^{(c)}=\frac{1}{n_c}\sum_{i\in\Pi_c}\boldsymbol{\phi}^{(i)}$, where n_c is the number of instances in Π_c). The ℓ_2-loss function is minimized iteratively by assigning each data instance i to the cluster c whose centroid $\bar{\boldsymbol{\phi}}^{(c)}$ is the nearest to $\boldsymbol{\phi}^{(i)}$ as

$$\pi(i)=\arg\min_c\left\|\boldsymbol{\phi}^{(i)}-\bar{\boldsymbol{\phi}}^{(c)}\right\|_2 . \quad (1)$$

Since neither $\boldsymbol{\phi}^{(i)}$ nor $\bar{\boldsymbol{\phi}}^{(c)}$ can be accessible explicitly, the square of the ℓ_2-norm in Eq. (1) is expanded in terms of entries from the kernel matrix K as

$$\left\|\boldsymbol{\phi}^{(i)} - \bar{\boldsymbol{\phi}}^{(c)}\right\|_2^2 = K_{ii} - \frac{2}{n_c}\sum_{a\in\Pi_c} K_{ia} + \frac{1}{n_c^2}\sum_{a,b\in\Pi_c} K_{ab}\,. \tag{2}$$

That expansion makes the computational complexity of finding the nearest centroid to each data instance of $O(n)$ and that of a single iteration over all data instance of $O(n^2)$. Further, an $O(n^2)$ space needs to be used to store the kernel matrix K. These quadratic complexities hinders applying the kernel k-means algorithm to large datasets.

Recent approximations have been proposed to allow using kernel k-means for large-scale datasets. In [4], it was suggested to restrict the clustering centroids to a subspace of data instances whose rank is at most l where $l \ll n$. That approximation was shown to significantly reduce the runtime and space complexity of the kernel k-means. It was also noticed by the authors that their method is equivalent to applying the original kernel k-means algorithm to the rank-l Nyström approximation [8] of the entire kernel matrix. The accuracy of the Nyström approximation is however determined by the value of l which has to be increased as the data size increases [8]. Later, Chitta et al. [5] exploited the Random Fourier Features (RFF) [14] to propose a fast algorithm for approximating the kernel k-means. However, that approximation inherits the limitation of the used RFF approach of being only applicable to shift-invariant kernels. RFF also requires representing the data instances into a vector space. Furthermore, the theoretical and empirical results of Yang et al. [16] showed that the kernel approximation accuracy of RFF-based methods depends on the properties of the eigenspectrum of the original kernel matrix which vary among different datasets. In our experiments, we show that the proposed approximation achieves superior clustering performance to the approximations of [4] and [5].

3 Approximate Nearest Centroid Embedding

Our embedding method is based on the results of Indyk [9] that showed that the ℓ_p-norm of a d-dimensional vector $\boldsymbol{v}$ can be estimated by means of p-stable distributions. Given a d-dimensional vector $\boldsymbol{r}$ whose components are i.i.d. samples drawn from a p-stable distribution over $\mathbb{R}$, the ℓ_p-norm of $\boldsymbol{v}$ is given by $||\boldsymbol{v}||_p = \alpha\mathbb{E}[|\sum_{i=1}^{d}\boldsymbol{v}_i\boldsymbol{r}_i|]$, where $\boldsymbol{v}_i$ denotes the i-th component of $\boldsymbol{v}$ and α is a positive constant. It is known that the standard Gaussian distribution $\mathcal{N}(0,1)$ is 2-stable [9] which means that it can be employed to compute the ℓ_2-norm of Eq. (1) as

$$||\boldsymbol{\phi} - \bar{\boldsymbol{\phi}}||_2 = \alpha\mathbb{E}[|\sum_{i=1}^{d}(\boldsymbol{\phi}_i - \bar{\boldsymbol{\phi}}_i)\boldsymbol{r}_i|]\,, \tag{3}$$

where d is the dimensionality of the space endowed by the used kernel function and the components $\boldsymbol{r}_i \sim \mathcal{N}(0,1)$. The expectation above can be approximated by the sample mean of multiple values for the term $|\sum_{i=1}^{d}(\boldsymbol{\phi}_i - \bar{\boldsymbol{\phi}}_i)\boldsymbol{r}_i|$ computed using m different vectors $\boldsymbol{r}$ each of which is denoted as $\boldsymbol{r}^{(j)}$. Thus, the ℓ_2-norm in Eq. (3) can be approximated as

$$||\boldsymbol{\phi} - \bar{\boldsymbol{\phi}}||_2 \approx \frac{\alpha}{m}\sum_{j=1}^{m}|\sum_{i=1}^{d}\left(\boldsymbol{\phi}_i\boldsymbol{r}_i^{(j)} - \bar{\boldsymbol{\phi}}_i\boldsymbol{r}_i^{(j)}\right)|\,. \tag{4}$$

Define two m-dimensional embeddings $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$ such that $\boldsymbol{y}_j = \sum_{i=1}^{d}\boldsymbol{\phi}_i\boldsymbol{r}_i^{(j)}$ and $\bar{\boldsymbol{y}}_j = \sum_{i=1}^{d}\bar{\boldsymbol{\phi}}_i\boldsymbol{r}_i^{(j)}$ or equivalently, $\boldsymbol{y}_j = \boldsymbol{\phi}^T\boldsymbol{r}^{(j)}$ and $\bar{\boldsymbol{y}}_j = \bar{\boldsymbol{\phi}}^T\boldsymbol{r}^{(j)}$. Eq. (4) can be expressed in terms of $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$ as

$$||\boldsymbol{\phi} - \bar{\boldsymbol{\phi}}||_2 \approx \frac{\alpha}{m}\sum_{j=1}^{m}|\boldsymbol{y}_j - \bar{\boldsymbol{y}}_j| = \frac{\alpha}{m}\left\|\boldsymbol{y} - \bar{\boldsymbol{y}}\right\|_1\,. \tag{5}$$

Since all of $\boldsymbol{\phi}$, $\bar{\boldsymbol{\phi}}$ and $\boldsymbol{r}^{(j)}$ are intractable to explicitly work with, our next step is to kernelize the computations of $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$. Without loss of generality, let $\mathcal{T}_j = \{\hat{\boldsymbol{\phi}}^{(1)}, \hat{\boldsymbol{\phi}}^{(2)}, ..., \hat{\boldsymbol{\phi}}^{(t)}\}$ be a set of t randomly chosen data instances embedded and centered into the kernel space (i.e. $\hat{\boldsymbol{\phi}}^{(i)} = \boldsymbol{\phi}^{(i)} - \frac{1}{t}\sum_{j=1}^{t}\boldsymbol{\phi}^{(j)}$).

According to the central limit theorem, the vector $\boldsymbol{r}^{(j)} = \frac{1}{\sqrt{t}} \sum_{\boldsymbol{\phi} \in \mathcal{T}_j} \boldsymbol{\phi}$ approximately follows a multivariate Gaussian distribution $\mathcal{N}(0, \Sigma)$ where Σ is the covariance matrix of the underlying distribution of all data instances embedded into the kernel space [12]. But, according to our definition of $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$, the individual components of $\boldsymbol{r}^{(j)}$ have to be independent and identically Gaussians. To fulfil that requirement, we make use of the fact that decorrelating the variables of a joint Gaussian distribution is enough to ensure that the individual variables are independent and marginally Gaussians. Using the whitening transform, $\boldsymbol{r}^{(j)}$ is decorrelated as

$$\boldsymbol{r}^{(j)} = \frac{1}{\sqrt{t}} \tilde{\Sigma}^{-1/2} \sum_{\phi \in \mathcal{T}^{(j)}} \phi \,, \tag{6}$$

where $\tilde{\Sigma}$ is an approximate covariance matrix estimated using a sample of l data instances (denoted as $\mathcal{L}$) embedded into the kernel space and centred as well.

With $\boldsymbol{r}^{(j)}$ defined as in Eq. (6), the computation of $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$ can be fully kernelized by following similar simplification steps as in [12]. Accordingly, $\boldsymbol{y}$ and $\bar{\boldsymbol{y}}$ can be computed as follows. Let $K_{\mathcal{L}\mathcal{L}}$ be the kernel matrix of $\mathcal{L}$ and define a centering matrix $H = I - \frac{1}{l}\boldsymbol{e}\boldsymbol{e}^T$ where I is an $l \times l$ identity matrix and $\boldsymbol{e}$ is a vector of all ones. Denote the inverse square root of the centered version of $K_{\mathcal{L}\mathcal{L}}$ as E.[1] The embedding of a data instance i is then given by

$$\boldsymbol{y}^{(i)} = f\left(\boldsymbol{\phi}^{(i)}\right) = R\Phi_{:\mathcal{L}}^T \phi^{(i)} = RK_{\mathcal{L}i} \,, \tag{7}$$

such that $\Phi_{:\mathcal{L}}$ is a $d \times l$ matrix of the elements of $\mathcal{L}$ and $K_{\mathcal{L}i}$ is an l-dimensional vector whose components are the kernel between i and all data instance in $\mathcal{L}$. R is an $m \times l$ matrix whose rows are computed as $R_{j:} = \boldsymbol{s}^T E H$, where $\boldsymbol{s}$ is an l-dimensional binary vector indexing t randomly chosen values from 1 to l for each j. Using Eq. (7) the embedding of the centroid of a cluster c is given by

$$\bar{\boldsymbol{y}}^{(c)} = f\left(\bar{\boldsymbol{\phi}}^{(c)}\right) = \frac{1}{n_c} R\Phi_{:\mathcal{L}}^T \sum_{i \in \Pi_c} \phi^{(i)} = \frac{1}{n_c} \sum_{i \in \Pi_c} R\Phi_{:\mathcal{L}}^T \phi^{(i)} = \frac{1}{n_c} \sum_{i \in \Pi_c} \boldsymbol{y}^{(i)} \,. \tag{8}$$

From Equations (1) and (5), it can be noticed that each data instance i can be assigned to an approximate nearest cluster using the embedding of i denoted as $\boldsymbol{y}^{(i)}$ and the embeddings of the current cluster centroids $\bar{\boldsymbol{y}}^{(c)}$ for $c = 1, 2, ..., k$, as

$$\tilde{\pi}(i) = \arg\min_c \left\| \boldsymbol{y}^{(i)} - \bar{\boldsymbol{y}}^{(c)} \right\|_1 \,. \tag{9}$$

Now, it can be noticed that Eq. (9) obviates the need to the expansion in Eq. (2) and instead, cluster assignments of Eq. (1) can be made only using pre-computed embeddings $\boldsymbol{y}^{(i)}$ for each data instance i. Furthermore, at the end of each iteration, Eq. (8) allows for computing updated embeddings of the new cluster centroids only using the embeddings of the data instances assigned to each cluster. Accordingly, Equations (8) and (9) suggest the following approximate kernel k-means algorithm. As a preprocessing step, an embedding $\boldsymbol{y}^{(i)}$ for each data instance i is computed using Eq. (7). Afterwards, a Lloyd-like k-means [13] iterations are performed over the embeddings, with the ℓ_2-distance replaced with the ℓ_1-distance, until convergence. The resulting cluster assignments represent an approximate clustering using the kernel k-means algorithm. In Appendix A., we outline the steps of the proposed algorithm.

Similar to the approximate kernel k-means approach presented in [4], APNC embedding requires computing an $l \times l$ kernel matrix $K_{\mathcal{L}\mathcal{L}}$ of a sample of l data instances. However, intuitively, we expect that APNC kernel k-means is capable of achieving significantly higher clustering accuracy with small values of l since the sample data instances are used to only estimate the covariance matrix of the underlying distribution. Given a fixed computing infrastructure, that property makes APNC kernel k-means able to handle larger datasets while providing reasonable clustering accuracy. Furthermore, after computing the embeddings of all data instances, the clustering step is done by a linear k-means variant which is easy to parallelize on distributed cloud infrastructures by sharing the embeddings of cluster centroids among all nodes and updating them after each iteration.[2]

[1] The kernel matrix over centered vectors can be calculated in terms of the kernel matrix over original vectors as $HK_{\mathcal{L}\mathcal{L}}H$. Its inverse square root can be computed as $\Lambda^{-1/2}V^T$ where, Λ and V are the eigenvalues and eigenvector matrices of $HK_{\mathcal{L}\mathcal{L}}H$ respectively.

[2] The details of the distributed implementation are to be included in an extended version of this paper.

4 Experiments

We evaluated the proposed algorithm by conducting experiments on medium and large-scale datasets. The medium-scale experiments were carried out on a single machine using four datasets denoted as **PIE**, **ImageNet-50k**, **USPS** and **MNIST**. **PIE** is a subset of $11,554$ face images under 68 classes [15]. **ImageNet-50k** is a subset of $50,000$ images under 164 classes, sampled from the $1,262,102$ images of the full ImageNet dataset used by Chitta et al. [4]. Both of **USPS** and **MNIST** are handwritten digits under 10 classe and their sizes are $9,298$ and $70,000$ respectively [3]. We used an RBF kernel for **PIE** and **ImageNet-50k**, a neural kernel for **USPS** and a polynomial kernel for **MNIST**. We used the same kernel parameters of [4]. Clustering accuracy is measured by the Normalized Mutual Information (NMI) between clustering labels and ground-truth labels.

In the medium-scale experiments, we compared our approach (*APNC*) to the approximate kernel k-means approach (*Approx KKM*) of [4] and the two Random Fourier Features based algorithms (*RFF*) and (*SV-RFF*) presented in [5]. For *APNC* and *Approx KKM* we used three different values for the number of samples l while fixing the parameter t in *APNC* to 40% of l and m to 1000. For a fair comparison, we set the number of fourier features used in *RFF* and *SV-RFF* to 500 to obtain 1000-dimensional embeddings as in *APNC*. Table 1 summarizes the average and standard deviation of the NMIs achieved in 20 different runs of each algorithm for each dataset using each value for l. Being limited to only shift-invariant kernels, both *RFF* and *SV-RFF* were only used for the datasets **PIE** and **ImageNet-50k**. It can be observed from Table 1 that the proposed method in this paper *APNC* is significantly superior to all the other methods especially when $l = 50$ which matches our intuition behind the scalability of APNC embedding. The results achieved by *APNC* method are also of smaller variations in clustering accuracy in most of the datasets which makes *APNC* more robust to the randomness and hence, more reliable. The poor performance of *RFF* and *SV-RFF* can be explained by the results of Yang et al. [16] that showed that for a fixed number of fourier features, the approximation accuracy of RFF-based methods are determined by the properties of the eigenspectrum of the kernel matrix being approximated.

Finally, we used MapReduce [6] to implement a parallel version of APNC kernel k-means on an Amazon EC2 [1] infrastructure of 20 nodes to cluster the full ImageNet dataset [4]. The last row in Table 1 shows the achieved NMIs. The best reported NMI for the full ImageNet dataset was 10.4% [4]. The table shows that, in addition to being easy to parallelize on cloud infrastructure, APNC kernel k-means is able to achieve superior clustering accuracy using a very small sample size.

Table 1: The NMIs of different kernel k-means approximations. For each dataset the best performing method(s) for each l according to a t-test (with 95% confidence) is(are) highlighted in bold.

Methods	$l = 50$	$l = 100$	$l = 300$
	PIE - 11K, RBF		
RFF	5.2 ± 0.12	5.2 ± 0.12	5.2 ± 0.12
SV-RFF	5.15 ± 0.11	5.15 ± 0.11	5.15 ± 0.11
Approx KKM	13.99 ± 0.6	14.66 ± 1.01	15.95 ± 0.83
APNC	$\mathbf{18.62 \pm 0.37}$	$\mathbf{19.5 \pm 0.38}$	$\mathbf{20.12 \pm 0.35}$
	ImageNet - 50K, RBF		
RFF	6.12 ± 0.04	6.12 ± 0.04	6.12 ± 0.04
SV-RFF	5.96 ± 0.06	5.96 ± 0.06	5.96 ± 0.06
Approx KKM	14.67 ± 0.25	15.12 ± 0.17	15.27 ± 0.15
APNC	$\mathbf{15.66 \pm 0.14}$	$\mathbf{15.78 \pm 0.14}$	$\mathbf{15.76 \pm 0.08}$
	USPS - 9K, Neural		
Approx KKM	37.60 ± 17.50	50.68 ± 11.28	$\mathbf{57.17 \pm 5.44}$
APNC	$\mathbf{52.88 \pm 7.25}$	$\mathbf{55.34 \pm 4.15}$	$\mathbf{58.22 \pm 0.87}$
	MNIST - 70K, Polynomial		
Approx KKM	19.07 ± 1.45	20.73 ± 1.30	22.38 ± 1.06
APNC	$\mathbf{23.00 \pm 1.57}$	$\mathbf{23.08 \pm 1.58}$	$\mathbf{23.86 \pm 1.82}$
	Full ImageNet - 1.2M, RBF		
Approx KKM	Best reported NMI is $\mathbf{10.4}$ [4].		
APNC	$\mathbf{11.29 \pm 0.03}$	$\mathbf{11.28 \pm 0.06}$	$\mathbf{11.30 \pm 0.01}$

Acknowledgment

We thank Radha Chitta and the authors of [4] for sharing their processed ImageNet dataset with us.

References

[1] *Amazon Elastic Compute Cloud (Amazon EC2)*. http://aws.amazon.com/ec2/.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin. A view of cloud computing. *Communication of the ACM*, 53(4):50–58, 2010.

[3] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), May 2011.

[4] R. Chitta, R. Jin, T. C. Havens, and A. K. Jain. Approximate kernel k-means: Solution to large scale kernel clustering. In *ACM SIGKDD KDD*, pages 895–903, 2011.

[5] R. Chitta, R. Jin, and A. K. Jain. Efficient kernel clustering using random Fourier features. In *IEEE ICDM*, pages 161–170, 2012.

[6] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[7] I. Dhillon, Y. Guan, and B. Kulis. Kernel k-means, spectral clustering and normalized cuts. In *ACM SIGKDD KDD*, pages 551–556, 2004.

[8] P. Drineas and M. W. Mahoney. On the Nyström Method for approximating a Gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6:2153–2175, 2005.

[9] P. Indyk. Stable distributions, pseudorandom generators, embeddings and data stream computation. In *Proceedings of the Symposium on Foundations of Computer Science*, 2000.

[10] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[11] H. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for MapReduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pages 938–948, Philadelphia, PA, USA, 2010. Society for Industrial and Applied Mathematics.

[12] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(6):1092–1104, 2012.

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

[14] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *NIPS*, pages 1177–1184, 2007.

[15] T. Sim, S. Baker, and M. Bsat. The cmu pose, illumination, and expression database. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(12):1615–1618, Dec. 2003.

[16] T. Yang, Y.-F. Li, M. Mahdavi, R. Jin, and Z.-H. Zhou. Nyström method vs random Fourier features: A theoretical and empirical comparison. In *NIPS*, pages 485–493, 2012.

Appendix A. APNC Kernel k-Means Algorithm

Algorithm 1 APNC Kernel k-Means

Input: Dataset $\mathcal{X}$ of n data instances, Kernel Function $\kappa(.,.)$,
APNC Parameters l, m, and t, Number of Clusters k
Output: Clustering Labels $\boldsymbol{l}$

$\mathcal{L} \leftarrow$ uniform sample of l data instances from $\mathcal{X}$
$K_{\mathcal{L}\mathcal{L}} \leftarrow \kappa(\mathcal{L}, \mathcal{L})$
$H \leftarrow I - \frac{1}{l} ee^T$
$K_{\mathcal{L}\mathcal{L}} \leftarrow HK_{\mathcal{L}\mathcal{L}}H$
$[V, \Lambda] \leftarrow eigen(K_{\mathcal{L}\mathcal{L}})$
$E \leftarrow \Lambda^{-1/2} V^T$
Initialize $R \leftarrow [0]_{m \times l}$
for $r = 1 : m$
    $\mathcal{T} \leftarrow$ select t unique values from 1 to p
    $R_{r:} = \sum_{v \in \mathcal{T}} E_{v:}$
end
$K_{:\mathcal{L}} \leftarrow \kappa(\mathcal{X}, \mathcal{L})$
$Y \leftarrow RHK_{:\mathcal{L}}^T$
$\boldsymbol{l} \leftarrow k$-Means($Y$, k, ℓ_1) *// Lloyd k-means with the ℓ_1-distance.*