2014

EFFICIENT APPROXIMATION FOR LARGE-SCALE KERNEL CLUSTERING ANALYSIS

Keng-Pei Lin
National Sun Yat-Sen University, kplin@mis.nsysu.edu.tw

Yu-Chen Yang
National Sun Yat-Sen University, ycyang@mis.nsysu.edu.tw

EFFICIENT APPROXIMATION FOR LARGE-SCALE KERNEL CLUSTERING ANALYSIS

Keng-Pei Lin, Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan, kplin@mis.nsysu.edu.tw

Yu-Chen Yang, Department of Information Management, National Sun Yat-sen University, Kaohsiung, Taiwan, ycyang@mis.nsysu.edu.tw

Abstract

Kernel k-means is useful for performing clustering on nonlinearly separable data. The kernel k-means is hard to scale to large data due to the quadratic complexity. In this paper, we propose an approach which utilizes the low-dimensional feature approximation of the Gaussian kernel function to capitalize a fast linear k-means solver to perform the nonlinear kernel k-means. This approach takes advantage of the efficiency of the linear solver and the nonlinear partitioning ability of the kernel clustering. The experimental results show that the proposed approach is much more efficient than a normal kernel k-means solver and achieves similar clustering performance.

Keywords: Data mining, Clustering Analysis, Kernel Methods

1 INTRODUCTION

Clustering is an important data mining task for grouping data into clusters where instances in a cluster are similar to each other and are dissimilar to those in other clusters. k-means (MacQueen, 1967) is a popular clustering algorithm for its simple to use and efficient implementations (Han and Kamber, 2006). Since k-means performs linear partitioning among instances, it is not applicable to linearly non-separable data. Kernel k-means (Dhillon, Guan and Kulis, 2004) is a kernelized variant of the k-means algorithm, which enables the nonlinear partitioning of data by applying the kernel trick (Schölkopf and Smola, 2002) to k-means algorithm. Kernel k-means captures the nonlinear property of the data by the implicitly nonlinear feature mapping induced by the kernel function.

In the iteration of the k-means algorithm, each instance $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, \dots, n$ is evaluated with all k clustering centers $\mathbf{m}_c$, $c = 1, \dots, k$ to find the nearest cluster center for new clustering assignment:

$$\arg\min_c |\mathbf{x}_i - \mathbf{m}_c|^2.$$

The expense of the nonlinear partitioning in the kernel k-means brought by applying the kernel trick is that the resulting clustering centers can only be represented as linear combinations of kernel evaluations with all instances but not explicit central points:

$$\mathbf{m}_c = \frac{1}{\sum_{i=1}^n U_{c,i}} \sum_{i=1}^n U_{c,i}\phi(\mathbf{x}_i) = \frac{1}{\sum_{i=1}^n U_{c,i}} \sum_{i=1}^n U_{c,i}K(\mathbf{x}_i,)$$

where $\phi()$ is a kernel-induced implicit feature mapping, $K(,)$ is a kernel function, and $U_{c,i} \in \{0, 1\}$, $c = 1, \dots, k$, $i = 1, \dots, n$ are the clustering assignments of the n instances to k clusters.

For normal k-means, the computing cost of each iteration in finding the nearest center for all instances increases linearly with the number of instances, i.e., the computing cost is $O(n)$. However, for the kernel k-means, the evaluation of each instance in each iteration already takes $O(n)$ due to the kernel evaluations with all instances, and therefore each iteration takes $O(n^2)$ computing cost in finding the nearest center point for all instances. With large-scale datasets, the number of instances can be very large. The heavy computational load of the kernel k-means algorithm causes it to converge slowly, and hence kernel k-means is still challenged to handle large-scale data.

In contrast, the normal k-means algorithm requires only $O(n)$ evaluations in each iteration since the center points can be explicitly derived. Compared to kernel k-means, the normal k-means can be much more efficient for handling large-scale datasets.

Despite the efficiency of the linear method of normal k-means for large-scale data, its applicability is constrained because it is only appropriate to the tasks with linearly separable data. An approach of leveraging the efficient linear method to perform the nonlinear partitioning of kernel k-means is explicitly representing the features induced by the nonlinear kernel function $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) \cdot \phi(\mathbf{y})$ and utilizing the explicit features $\phi(\mathbf{x}_i)$, $i = 1, \dots, n$ as the input to the normal k-means solver. If the dimensions of the explicit features are not too big, it can be very efficient to perform kernel k-means in this way. For example, the work of (Chitta, Jin and Jain, 2012) utilizes random Fourier features (Rahimi and Recht, 2007) to approximate the explicit features of the Gaussian kernel function to speedup kernel k-means. However, the random Fourier features are dense and a large number of features are needed to reduce variations in the approximation. The large number of features lowers the efficiency of the normal k-means solver and requires much memory space. If there are not enough number of random Fourier features, the large variation will degrade the precision of approximation which results in poor performance.

In this paper, we utilize Taylor polynomial to approximate the commonly used Gaussian kernel function (Lin and Chen, 2011; Yang, Duraiswami and Davis, 2004) with a normal k-means solver to perform kernel k-means. The Gaussian kernel function can be represented as a Taylor series of features, and then a low-dimensional approximating feature mapping is obtained by taking a low-degree Taylor polynomial. The explicitly listed approximated features of the Gaussian kernel function in the low-degree Taylor polynomial is utilized as input to a normal k-means solver. This takes advantage of the efficiency of the linear method and achieves close clustering performance to the nonlinear kernel k-means. Unlike the uniform approximation of utilizing the random Fourier features, the Taylor polynomial of features are compact since the important components are concentrated in low-order terms of the Taylor series. Furthermore, if the original features are sparse, the Taylor polynomial of features are also sparse. Therefore, only a few number of features in the low-order terms of the Taylor series are able to achieve good approximating precision. By approximating the feature mapping of the Gaussian kernel function with low-dimensional features and leveraging the efficiency of linear solvers, we can perform large-scale nonlinear clustering by kernel k-means efficiently. The experimental results show that the proposed method is much more efficient than the kernel k-means and achieves similar clustering performance.

The rest of this paper is organized as follows: In Section 2, we briefly survey related works. Then in Section 3, we discuss kernel k-means and the Taylor series features of Gaussian kernel function. In Section 4, we describe the approach for utilizing the approximating features in the Taylor polynomial with a linear solver to efficiently performing kernel k-means clustering. Section 5 shows the experimental results, and we conclude the paper in Section 6.

2 RELATED WORK

Kernel clustering method such as kernel k-means is hard to scale to large data due to the quadratic computational complexity and memory requirement of storing the kernel matrix. A popular approach to tackle the scalability problem of the kernel clustering is based on the Nyström approximation (Williams and Seeger, 2000) such as the work of (Chitta, Jin, Havens and Jain, 2011), which approximates the full kernel matrix with a low rank rectangular sub matrix of the full kernel matrix to perform the kernel k-means.

The work of (Chitta et al., 2012) employs random Fourier features (Rahimi and Recht, 2007), which uniformly approximates the implicit feature mapping of the Gaussian kernel function to leverage the efficient linear k-means solver. However, the random Fourier features are dense and a large number of random Fourier features are required since too few features will have very large variation in the approximation.

3 BACKGROUND

In this section, we discuss the kernel k-means and Taylor polynomial feature approximation of the Gaussian kernel function.

3.1 Kernel k-Means

The k-means is a partitioning method for clustering analysis. Given a user specified number of clusters k, the k-means algorithm partitions a set of instances $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ into k clusters. The *kernel trick* can be applied to k-means by replacing the dot product computations between instances with the kernel evaluations. The *kernel function* $k(\mathbf{x}_i, \mathbf{x}_j)$ implicitly maps $\mathbf{x}_i$ and $\mathbf{x}_j$ into a high-dimensional feature space and computes their dot product there. By applying the kernel trick to k-means algorithm, the data are implicitly mapped into the kernel induced high-dimensional feature space to find clusters. The kernel k-means augments k-means algorithm to partition linearly non-separable data to clusters.

Let $U_{k\times n}$ be the clustering membership matrix where $U_{c,i} \in \{0, 1\}$, $c = 1, \ldots, k$, $i = 1, \ldots, n$ denotes the clustering assignment of each instance. Let $\phi(\mathbf{x})$ denote the feature mapped instance $\mathbf{x}$ in the kernel-induced feature space. The objective of the kernel k-means is

$$
\begin{aligned}
&\arg\min_U \sum_{c=1}^{k}\sum_{i=1}^{n} U_{c,i}||\mathbf{m}_c - \phi(\mathbf{x}_i)||^2\\
&= \arg\min_U \sum_{c=1}^{k}\sum_{i=1}^{n} U_{c,i}\\
&[\frac{1}{\sum_{h=1}^{n} U_{c,h}\sum_{g=1}^{n} U_{c,g}}\sum_{h=1}^{n}\sum_{g=1}^{n} U_{c,h}U_{c,g}\phi(\mathbf{x}_h)\cdot\phi(\mathbf{x}_g)\\
&-\frac{2}{\sum_{h=1}^{n} U_{c,h}}\sum_{h=1}^{n} U_{c,h}\phi(\mathbf{x}_h)\cdot\phi(\mathbf{x}_j) + \phi(\mathbf{x}_j)\cdot\phi(\mathbf{x}_j)].
\end{aligned}
$$

The dot product of feature mapped instances is obtained from the kernel matrix $K_{n\times n}$ where

$$K_{h,g} = k(\mathbf{x}_h, \mathbf{x}_g) = \phi(\mathbf{x}_h)\cdot\phi(\mathbf{x}_g),\ h = 1,\ldots,n,\ g = 1,\ldots,n.$$

The kernel k-means can be implemented by an greedy approach such as Lloyd algorithm (Lloyd, 1982) for normal k-means: In each iteration, update the clustering membership matrix U for each instance $\mathbf{x}_i$ according to its closest center point $\mathbf{m}_c$ in the feature space

$$\arg\min_c ||\mathbf{m}_c - \phi(\mathbf{x}_i)||^2$$

until the matrix U does not change or the user-specified maximum number of iterations is reached. The square sum error is reduced in each iteration. Since the center points $\mathbf{m}_c$, $c = 1,\ldots,k$ are in the feature space and can not be derived explicitly, the initial set of random k center points are obtained by randomly initialize the indicating values of the clustering membership matrix U. The update of the clustering membership matrix U in each iteration implies the recalculation of center points $\mathbf{m}_c$, $c = 1,\ldots,k$ in the feature space.

3.2 Taylor Polynomial Approximation of Gaussian Kernel Function

The Gaussian kernel function is an exponential function of the Euclidean distance between two instances:

$$K(\mathbf{x},\mathbf{y}) = \exp(-g||\mathbf{x}-\mathbf{y}||^2)$$

where $g > 0$ is a user-specified kernel parameter. The Gaussian kernel function induces infinite-dimensional implicit feature mapping (Schölkopf and Smola, 2002).

The feature mapping of the Gaussian kernel function can be represented as a Taylor series of monomial features $K(\mathbf{x},\mathbf{y}) = \Phi_G(\mathbf{x})\cdot\Phi_G(\mathbf{y})$(Lin and Chen, 2011; Yang et al., 2004). For an instance $\mathbf{x}$, the infinite-dimensional feature mapping induced by the Gaussian kernel function can be

$$\Phi_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\left[\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{x})|d = 0,\ldots,\infty\right]$$

where $\Phi_d(\mathbf{x})$, $d = 1,\ldots,\infty$ is degree-d monomial features of $\mathbf{x}$ (Smola, Schölkopf and Müller, 1998) which is defined as

$$
\begin{aligned}
&\Phi_d(\mathbf{x}) =\\
&[\sqrt{\frac{d!}{\prod_{i=1}^{N} m_{k,i}!}}\prod_{i=1}^{N} x_i^{m_{k,i}}|\forall\mathbf{m}_k \in \mathbb{N}^N \text{ with } \sum_{i=1}^{N} m_{k,i} = d]
\end{aligned}
$$

Input: Instances $\mathbf{x}_i \in \mathbb{R}^N$, $i = 1, \ldots, n$, approximation degree d_u, Gaussian kernel parameter g.
Output: Clustering assignment matrix $U_{k\times n}$.

```
For each instance $\mathbf{x}_i$, apply the degree-$d_u$ $\bar{\Phi}_G$ feature mapping with kernel parameter $g$ to obtain $\bar{\Phi}_G(\mathbf{x}_i), i = 1, \ldots, n$.
Run a normal linear $k$-means algorithm on $\bar{\Phi}_G(\mathbf{x}_i)$, $i = 1, \ldots, n$ to generate the clustering assignment matrix $U$.
```

Figure 1: *Approximately performing the kernel k-means by Taylor polynomial features with a linear k-means solver.*

Each $\mathbf{m}_k$ corresponds to a dimension of degree-d monomial features, and there are totally $\binom{n+d-1}{d}$ features.

From the Taylor approximation, the Taylor series $\Phi_G(\mathbf{x})$ can be estimated by a low-degree Taylor polynomial. By keeping only the low-order terms of the Taylor series, we obtain a low-dimensional approximated feature mapping of the Gaussian kernel function

$$\bar{\Phi}_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\left[\sqrt{\frac{(2g)^d}{d!}}\Phi_d(\mathbf{x})|d = 0, \ldots, d_u\right]$$

where d_u is a user-specified approximation degree. The dimensionality of $\bar{\Phi}_G(\mathbf{x})$ for $\mathbf{x} \in \mathbb{R}^N$ is $\sum_{d=0}^{d_u}\binom{N+d-1}{d} = \binom{N+d_u}{d_u}$. The higher the d_u is, the closer to the original Gaussian kernel function the approximation gets. It is noted that the exponential function can be precisely approximated by a low-degree Taylor polynomial if the evaluating point is not too far from the defined point.

An example of a degree-2 Taylor polynomial approximated feature mapping with two-dimensional instances for Gaussian kernel function is as follows:

$$\begin{aligned}&\bar{\Phi}_G(\mathbf{x}) = \exp(-g||\mathbf{x}||^2)\\ &[1, \sqrt{2g}x_1, \sqrt{2g}x_2, \sqrt{\frac{(2g)^2}{2!}}x_1^2, \sqrt{2\frac{(2g)^2}{2!}}x_1x_2, \sqrt{\frac{(2g)^2}{2!}}x_2^2].\end{aligned}$$

Then the Gaussian kernel function can be approximately computed by

$$K(\mathbf{x}, \mathbf{y}) \approx \bar{\Phi}_G(\mathbf{x}) \cdot \bar{\Phi}_G(\mathbf{y}).$$

4 EFFICIENT APPROXIMATION OF KERNEL K-MEANS

With the low-dimensional Taylor polynomial feature mapping $\bar{\Phi}_G$ to approximately compute the Gaussian kernel function, we can utilize an efficient normal k-means solver with the approximated feature mapped instances to perform kernel k-means. The instances are explicitly and nonlinearly mapped, and then the normal k-means algorithm partitions the data in the feature space.

Figure 1 shows the algorithm for performing kernel k-means by the Taylor polynomial features with a linear k-means solver. First, all instances are transformed to Taylor polynomial features by a user-specified approximating degree. Then a linear k-means solver is utilized to compute explicit clustering central points in each iteration and update the clustering membership assignment according to the closest central point of each feature mapped instance.

Compared to utilizing the uniformly random Fourier features (Rahimi and Recht, 2007) for approximation, the Taylor polynomial-based approximation of the Gaussian kernel function is non-uniform. Significant information is concentrated in low-degree terms due to the property of the Taylor approximation. Hence we can utilize a few low-degree features to precisely approximate the infinite-dimensional feature mapping of the Gaussian kernel function. Then the kernel k-means can be approximately performed via the linear k-means solver. This approach leverages the fast linear solver to perform the nonlinear kernel-based algorithms.

5 EXPERIMENT

In the experiments, we compare the clustering performance of the proposed Taylor polynomial-based kernel k-means with a normal kernel k-means algorithm to evaluate the efficiency and the performance of the proposed scheme. All the programs are implemented in Matlab. Our testbed features an Intel Core i5-3210M CPU at 2.5 GHz and 16GB RAM. The datasets utilized in the experiments are available from the UCI machine learning repository (Bache and Lichman, 2013). We choose two datasets for experiments: Pen Digits and Forest Cover Type. Since the normal kernel k-means algorithm consumes lots of memory to store the full kernel matrix, we sampled $40,000$ instances from the Forest Cover Type dataset to enable the execution of the normal kernel k-means algorithm. All feature values are normalized to $[0, 1]$ in pre-processing steps to prevent the effect that features in a large value range may dominate smaller ones. The statistics of the datasets are shown in Table 1.

Dataset	Instances	Features	Classes
Pen	$10,992$	16	10
Forest	$40,000$	54	7

Table 1: *Dataset Statistics.*

In the whole experiments, we use the degree-2 Taylor polynomial feature mapping which results in low-dimensional features and the instances can be transformed very fast. In both the normal kernel k-means and the Taylor polynomial-based kernel k-means, the Gaussian kernel parameter g is set to the inverse of dimensions, i.e., $g = \frac{1}{16}$ for Pen Digits and $g = \frac{1}{54}$ for the Forest Cover Type. There are heuristics to set the Gaussian kernel parameter (Alzate and Suykens, 2010; Perona and Zelnik-Manor, 2004). We simply use the rule-of-thumb inverse of dimensions to set the Gaussian kernel parameter.

5.1 Clustering Performance

The labels provided in the datasets are utilized as the external criteria to compare the clustering results. We adopt the normalized mutual information (NMI) (Strehl and Ghosh, 2002) to evaluate the clustering performance. The NMI measure has values in $[0, 1]$. A value 0 means that the two clusterings are independent and share no information, and a value 1 indicates that the two clusterings are the same. The experimental results are shown in Table 5.1. All the results are average values of 10 random repetitions and the standard deviations are shown in the parentheses. The experimental results show that the Taylor polynomial kernel k-means achieves similar clustering performance to a normal kernel k-means algorithm.

5.2 Clustering Efficiency

The comparison of execution time is shown in Table 3. It is seen that the Taylor polynomial kernel k-means consumes much less time than the normal kernel k-means algorithm due to the efficiency of the linear k-means solver.

6 CONCLUSION

In this paper, we propose the Taylor polynomial-based kernel k-means algorithm which precisely approximates the infinite-dimensional implicit feature mapping of the Gaussian kernel function by low-

Dataset	Normal kernel k-means	Taylor polynomial kernel k-means
Pen Digits	0.6775 (0.0165)	0.6773 (0.0145)
Forest Covery	0.1339 (0.0180)	0.1384 (0.0183)

Table 2: *Clustering performance (NMI) comparison.*

Dataset	Normal kernel k-means	Taylor polynomial kernel k-means
Pen	25.6	1.5
Forest	116.2	16.4

Table 3: *Clustering efficiency comparison (in seconds).*

dimensional features, and leverage the linear k-means solver to approximately perform the nonlinear kernel k-means. The experimental results show that the Taylor polynomial-based kernel k-means algorithm can achieve similar clustering performance like a normal kernel k-means algorithm and is much more efficient.

References

Alzate, C. and Suykens, J. (2010), 'Multiway spectral clustering with out-of-sample extensions through weighted kernel PCA', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(2), 335–347.

Bache, K. and Lichman, M. (2013), 'UCI machine learning repository'. `http://archive.ics.uci.edu/ml`.

Chitta, R., Jin, R., Havens, T. C. and Jain, A. K. (2011), Approximate kernel k-means: Solution to large scale kernel clustering, *in* 'Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery in data mining (KDD)'.

Chitta, R., Jin, R. and Jain, A. K. (2012), Efficient kernel clustering using random fourier features, *in* 'Proceedings of the 12th IEEE International Conference on Data Mining (ICDM)'.

Dhillon, I. S., Guan, Y. and Kulis, B. (2004), Kernel k-means, spectral clustering and normalized cuts, *in* 'Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery in data mining (KDD)'.

Han, J. and Kamber, M. (2006), *Data Mining: Concepts and Techniques*, Morgan Kaufmann.

Lin, K.-P. and Chen, M.-S. (2011), Efficient kernel approximation for large-scale support vector machine classification, *in* 'Proceedings of the 11th SIAM International Conference on Data Mining (SDM)'.

Lloyd, S. P. (1982), 'Least squares quantization in PCM', *IEEE Transactions on Information Theory* **28**(2), 129–137.

MacQueen, J. (1967), Some methods for classification and analysis of multivariate observations, *in* 'Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability'.

Perona, P. and Zelnik-Manor, L. (2004), Self-tuning spectral clustering, *in* 'Advances in Neural Information Processing Systems 17 (NIPS)'.

Rahimi, A. and Recht, B. (2007), Random features for large-scale kernel machines, *in* 'Advances in Neural Information Processing Systems 20 (NIPS)'.

Schölkopf, B. and Smola, A. J. (2002), *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press.

Smola, A. J., Schölkopf, B. and Müller, K.-R. (1998), 'The connection between regularization operators and support vector kernels', *Neural Networks* **11**, 637–649.

Strehl, A. and Ghosh, J. (2002), 'Cluster ensembles - a knowledge reuse framework for combining multiple partitions', *Journal of Classification* **3**, 583–617.

Williams, C. and Seeger, M. (2000), Using the Nyström method to speed up kernel machines, *in* 'Advances in Neural Information Processing Systems 13 (NIPS)', MIT Press, pp. 682–688.

Yang, C., Duraiswami, R. and Davis, L. (2004), Efficient kernel machines using the improved fast gauss transform, *in* 'Advances in Neural Information Processing Systems 17 (NIPS)'.