KMEANS and Kernel KMEANS

A comparative study of classical and kernel kmeans for data clustering.

Ezukwoke K.I[1], Zareian S.J[2]

[1, 2] Department of Computer Science
Machine Learning and Data Mining
{ifeanyi.ezukwoke, samaneh.zareian.jahromi}@etu.univ-st-etienne.fr
University Jean Monnet, Saint-Etienne, France

Abstract

Kmeans is a simple yet efficient unsupervised clustering algorithm. In this paper we study classical and different kernel Kmeans. We take an experimental analysis on datasets including (moon, circle, classification and iris dataset). We briefly introduce multiple kernel learning and its applications in kernel kmeans. We conclude by expanding on the performance of both algorithms to see which is most suited for data clustering.

Keywords
Partition clustering, K-Means, kernel K-Means.

1 INTRODUCTION

K-Means clustering is a fast, robust, and simple algorithm that gives reliable results when data sets are distinct or well separated from each other in a linear fashion. It is best used when the number of cluster centers, is specified due to a well-defined list of types shown in the data. However, it is important to keep in mind that K-Means clustering may not perform well if it contains heavily overlapping data, if the Euclidean distance does not measure the underlying factors well, or if the data is noisy or full of outliers. [1]

2 PARTITION CLUSTERING

Clustering is the task of gathering samples into groups of similar samples according to some predefined similarity and they find applications in *data compression, data summurization for recommender systems, similarity grouping of web search result, stock indices and customer profiles.*

KMeans falls under the clustering category called **partition clustering**. This approach used a technique of splitting the datasets into subgroups of $k-clusters$ and iteratively tries to find the best $k-cluster$ that best explains the partition of a data.

2.1 KMEANS

The objective of traditional clustering methods is to partition training vectors by using similarity criteria applied on the basis of Euclidean metrics. More precisely, the Euclidean distance or inner product is used to measure the similarity between the training vectors in the original vector space,

[1]source code for project is available on github

$\{x_i, i = 1, \dots, N\}$. The objective function we try to minimize in KMeans is give by

$$argmin_{w_k} \left\{ \sum_{k=1} K \sum_{X_t \in C_k} \|x_t - \mu_k\|^2 \right\} \quad (1)$$

Where μ_k denotes the mean of *kth* cluster's centroid. In an optimal K-means solution, the centroid, say μ_k is associated with a training vector x_j that yields the minimum distance among all the centroids.

$$\mu_k = \frac{1}{C_k} \sum_{i \in C_k} x_i \quad (2)$$

Lloyd algorithm is one of the most well used classical kmeans algorithm [1].

Algorithm 1: Classical K-Means (Lloyd's) algorithm

```
Input  : $X, \mu^r$ random center
         initialization
Output : $\mu_k \in C_k$
1  begin
2      while not converged do
3          $C_1, \dots, C_k \leftarrow \phi$;
4          for $i \in 1, \dots, n$ do
5              $\arg\min \sum_{c=1}^{k} \sum_{x_i \in C_k} ||x_i - \mu_k||^2$
6          end
7          for $j \in 1, \dots, k$ do
8              $\mu_j \leftarrow \frac{1}{C_k} \sum_{i \in C_j} x_i$
9          end
10     end
11 end
```

2.2 Kernel KMEANS

Kernel Kmeans is a non-linear version of the classical kmeans algorithm. Several method have been proposed and used over the years like the spectral version [2]. The data x_i is projected into a higher dimensional subspace $\phi(x_i)$ where we are only concerned in the dot products of the feature vectors. The distance in the new space becomes

$$\|\phi(x_i) - \phi(\mu_j)\|^2 \quad (3)$$

Where

$$\phi(\mu_k) = \frac{1}{C_k} \sum_{i \in C_k} \phi(x_i) \quad (4)$$

By expanding the above equation we have

$$\kappa(x_i, x_i) + \kappa(\mu_j, \mu_j) - 2\kappa(x_i, \mu_j) \quad (5)$$

Algorithm 2: Kernel K-Means clustering algorithm

```
Input  : $\kappa$
Output : $C_k$
1 begin
2     while not converged do
3         Compute the distance of each
            point in $\phi(x)$ from center $\mu$
            $\arg\min \sum_{c=1}^{k} \sum_{x_i \in C_k} ||\phi(x_i) - \phi(\mu_j)||^2$
4     end
5 end
```

2.3 Kernels

We introduce the commonly used kernels and a brief overview of Multiple kernels used.

- **Linear kernel**

$$\kappa(x_i, x_j) = \mathbf{x}_i \mathbf{x}_j^T \quad (6)$$

- **Polynomial kernel**

$$\kappa(x_i, x_j) = (\mathbf{x}_i \mathbf{x}_j^T + c)^d \quad (7)$$

  where $c \geq 0$ and d is the degree of the polynomial usually greated than 2.

- **RBF(Radial Basis Function) kernel**
  Sometimes referred to as the **Guassian kernel.**

$$\kappa(x_i, x_j) = \exp(-\gamma ||\mathbf{x}_i - \mathbf{x}_j||^2) \quad (8)$$

  where $\gamma = \frac{1}{2\sigma^2}$.

- **Sigmoid kernel**

$$\kappa(x_i, x_j) = \tanh(\gamma \mathbf{x}_i \mathbf{x}_j^T + c) \quad (9)$$

  where $c \geq 0$ and $\gamma = \frac{1}{2\sigma^2}$.

- **Cosine kernel**

$$\kappa(x_i, x_j) = \frac{\mathbf{x}_i \mathbf{x}_j^T}{||\mathbf{x}_i|| ||\mathbf{x}_j||} \quad (10)$$

2.4 Multi-kernel

The reason behind the use of multiple kernel is similar to the notion of multiclassification, where cross-validation is used to select the best performing classifier [3]. By using multiple kernel, we hope to learn a different similar in the kernel space not easily observed when using single kernel.

We can prove from **Mercer's Theorem** that a kernel is **Positive Semi-Definite (PSD)** if $u^T\kappa(x_i, x_j)u \geq 0$. Hence by performing arithmetic or any mathematical operation on two or more kernel matrix, we obtain a new kernel capable of exploiting different property or similarities of training data.

Given a kernel κ, we prove that κ is PD if

$$\langle u, \kappa u\rangle \geq 0 \tag{11}$$

Proposition: *A symmetric function κ: $\chi \rightarrow \mathbb{R}$ is positive definite if and only if $\langle u, \kappa u\rangle \geq 0$*
Proof:
Suppose that κ is a kernel which is the inner product of the mapping functions $\big\langle \phi(x_i)\phi(x_j)\big\rangle$. κ is a kernel if its inner product are positive and the solution of $\kappa u = \lambda u$ gives non-negative eigenvalues.

So that,

$$\langle u, \kappa u\rangle = \sum_{i=1}^{N} u_i \cdot (\kappa u_i) \tag{12}$$

$$= \sum_{i=1}^{N} u_i \sum_{j=1}^{N} \big\langle \phi(x_i)\phi(x_j)_{\mathcal{H}}\big\rangle\, u_j \tag{13}$$

Where $\mathcal{H}$ represent the Hilbert space we project the kernel [?].

$$= \left\langle \sum_{i=1}^{N}\sum_{j=1}^{N} u_i\phi(x_i), u_j\phi(x_j)\right\rangle_{\mathcal{H}} \tag{14}$$

$$\langle u, \kappa u\rangle = \left\|\sum_{i=1}^{N} u_i\phi(x_i)\right\|_{\mathcal{H}}^{2} \geq 0 \tag{15}$$

Therefore κ is positive definite.

Using this property of the kernel κ we introduce multiple kernel combination as follow

- **LinearRBF**
  Here we combine two kernels, precisely **Linear and RBF kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{linrbf}} = \kappa(x_i, x_j) \times \kappa(x_i, x_l) \tag{16}$$

- **RBFPoly**
  Here we combine **RBF and Polynomial kernel** using their inner product.

$$\hat{\mathbf{K}}_{\mathbf{rbfpoly}} = \kappa(x_i, x_j) \times \kappa(x_i, x_l) \tag{17}$$

- **EtaKernel**
  The **EtaKernel** is a composite combination of **LinearRBF, RBFPoly and RBFCosine** and it is given by

$$\hat{\mathbf{K}}_{\mathbf{etarbf}} = \hat{\mathbf{K}}_{\mathbf{linrbf}} \times \hat{\mathbf{K}}_{\mathbf{rbfpoly}} + \hat{\mathbf{K}}_{\mathbf{rbfpoly}} \times \hat{\mathbf{K}}_{\mathbf{rbfcosine}}$$

3 EXPERIMENTAL RESULT

We experiment on benchmark datasets including moons, circle, classification and iris datasets.

3.1 Dataset

We generate a total of 1000 samples for each datasets and test out our algorithm using different configurations.

We begin with a high value for gamma.

3.2 Performance analysis

We analyse the performance of classical kmeans and compare with kernel kmeans with different settings of gamma (γ) and polynomial degree d. We compare their performances using the randindex as an evaluation metric, since the true labels of the datasets used are known. We visualize the result for the different settings of γ and d used.

3.2.1 Evalaution metric

The Rand index is a clustering metric used when the labels of the original dataset are known. Its functions similar to accuracy (for classification). The formula for computing rand index is given by

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \tag{18}$$

where TP is true positives, TN is true negatives, FP is false positives and FN is false negatives.

3.2.2 Gamma $\gamma = 0.01$ and polynomial degree $d = 2$

We observe result of kmeans for small gamma ($\gamma = 0.01$). We notice from 1 starting with circle dataset. For a small gamma and polynomial degree of $d = 2$, all kernel behave like the classic kmeans except for **laplace and eta kernel**. the etakernel behaves cyclical almost for all observed result as we will see later. The best performing kernel for circle dataset is laplace kmeans with 89% randindex.

Similarly for a moon dataset **laplace and linear (classic) kernel** are the best performing kernels, both with randindex of 63%. classic kmeans is best performing for classification dataset while laplace kernel is best peforming for Iris dataset.

From table 2, table of running time, classic kmeans and sigmoid kernel kmeans are the faster running algorithms. Classic kmeans outperforms other kernel versions for moon and iris datasets, while sigmoid kernel outperforms all others for circle and classification datasets.

3.2.3 Gamma $\gamma = 1$ and polynomial degree $d \geq 2$

We observe an improvement in rand index for non-linearly seperable data (*moon, circle and classification*). Rand index for rbf kernel on moon dataset is increased from 69% to 74%, making it the best performing kernel for clustering moon dataset. On circle dataset, laplace kernel is supreme over other kernels with rbf kernel falling just 1% behind.

Sigmoid kernel is the best performing kernel for classification dataset and classic kmeans is the best performing for iris dataset.

In terms of time, classic kmeans and sigmoid kernel kmeans are the faster of all kernels used. Linear or classic kmeans is fastest in clustering moon, classification and iris datasets while sigmoid is fastest in clustering circle dataset.

3.2.4 Gamma $\gamma = 5$ and polynomial degree $d \geq 2$

We examine the performance of kmeans for a higher gamma ($\gamma = 5$)and polynomial degree ($d = 5, d = 7$) for moon and circle datasets respectively and $d = 3, d = 2$ for classification and iris datasets respectively.

We observe that **rbf kernel** remains the best performing kernel for moon and circle datasets with a respectively 68% and 100% rand index. sigmoid kernel is best performing kernel for classification dataset while linear kernel is best performing for iris dataset.

We observe a noticeable random change in rand index for classic kmeans, this is due to the non-deterministic nature of selecting centroids at algorithm start.

Sigmoid kernel has the fastest running time for moon dataset, rbf for circle dataset, polynomial for classification dataset and linear for iris dataset.

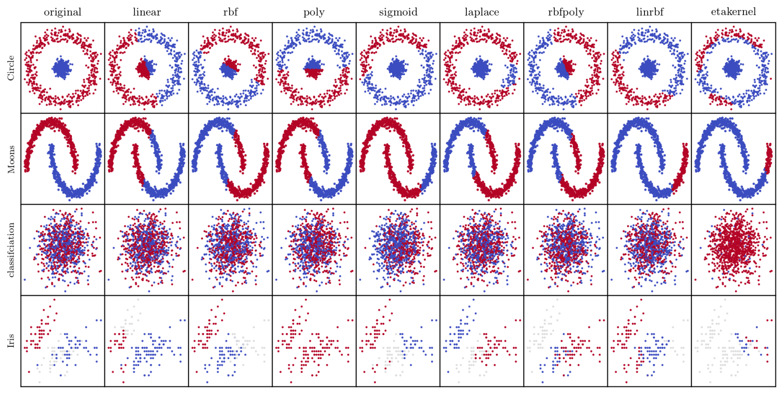

Figure 1: Classic and kernel kmeans on *circle, moons, classification and iris datasets.* Moon dataset ($\gamma = 0.01$ and $d = 3$), Circle dataset ($\gamma = 1$ and $d = 3$), Classification dataset ($\gamma = 0.01$ and $d = 2$), Iris dataset ($\gamma = 0.01$ and $d = 2$)

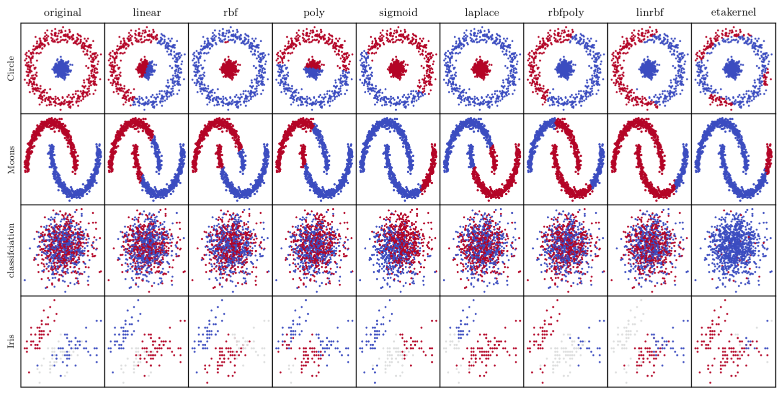

Figure 2: Classic and kernel kmeans on *circle, moons, classification and iris datasets.* Moon dataset ($\gamma = 1$ and $d = 3$), Circle dataset ($\gamma = 1$ and $d = 3$), Classification dataset ($\gamma = 1$ and $d = 2$), Iris dataset ($\gamma = 0.1$ and $d = 2$)

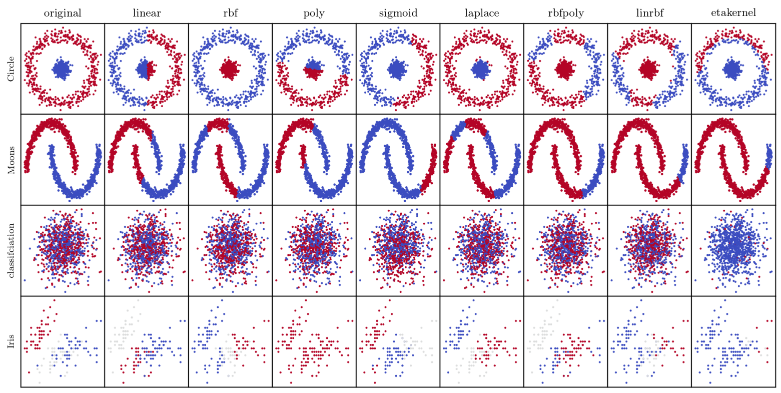

Figure 3: Classic and kernel kmeans on *circle, moons, classification and iris datasets.* Moon dataset ($\gamma = 5$ and $d = 5$), Circle dataset ($\gamma = 5$ and $d = 7$), Classification dataset ($\gamma = 2$ and $d = 3$), Iris dataset ($\gamma = 1$ and $d = 2$)

RandIndex (%)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**63**	59	56	61	63	56	62	51
Circle	51	50	59	**50**	89	59	51	58
Classification	80	62	49	**61**	66	63	65	49
Iris	**71**	87	82	32	88	70	87	48

Table 1: RandIndex(in %) for Moon dataset ($\gamma = 0.01$ and $d = 3$), Circle dataset ($\gamma = 0.01$ and $d = 3$), Classification dataset ($\gamma = 0.01$ and $d = 2$), Iris dataset ($\gamma = 0.01$ and $d = 2$)

Running Time (*secs*)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**0.07**	0.15	0.11	0.09	0.19	0.18	0.07	0.24
Circle	0.20	0.17	0.14	**0.06**	0.07	0.17	0.2	0.27
Classification	0.14	0.49	0.16	**0.2**	0.31	0.36	0.3	0.56
Iris	**0.001**	0.002	0.002	0.004	0.002	0.003	0.002	0.004

Table 2: Running time(in *secs*) for Moon dataset ($\gamma = 0.01$ and $d = 3$), Circle dataset ($\gamma = 0.01$ and $d = 3$), Classification dataset ($\gamma = 0.01$ and $d = 2$), Iris dataset ($\gamma = 0.01$ and $d = 2$). the bold numbers indicate faster running time.

RandIndex (%)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**63**	74	56	63	72	53	59	51
Circle	50	99	60	**50**	100	61	65	56
Classification	51	49	50	**67**	49	49	49	49
Iris	**86**	88	84	58	71	81	77	45

Table 3: RandIndex(in %) for Moon dataset ($\gamma = 1$ and $d = 3$), Circle dataset ($\gamma = 1$ and $d = 3$), Classification dataset ($\gamma = 1$ and $d = 2$), Iris dataset ($\gamma = 0.1$ and $d = 2$)

Running Time (*secs*)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**0.02**	0.22	0.11	0.06	0.16	0.19	0.26	0.25
Circle	0.10	0.18	0.10	**0.04**	0.07	0.27	0.19	0.34
Classification	0.14	0.16	0.18	**0.26**	0.15	1.10	1.19	0.62
Iris	**0.001**	0.002	0.003	0.006	0.015	0.002	0.003	0.05

Table 4: Running time(in *secs*) for Moon dataset ($\gamma = 5$ and $d = 5$), Circle dataset ($\gamma = 5$ and $d = 7$), Classification dataset ($\gamma = 2$ and $d = 3$), Iris dataset ($\gamma = 1$ and $d = 2$). the bold numbers indicate faster running time.

RandIndex (%)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**63**	68	56	63	58	54	51	51
Circle	52	100	59	**50**	99	64	60	55
Classification	68	49	52	**87**	50	49	49	49
Iris	**88**	78	87	87	91	81	79	45

Table 5: RandIndex(in %) for Moon dataset ($\gamma = 5$ and $d = 5$), Circle dataset ($\gamma = 5$ and $d = 7$), Classification dataset ($\gamma = 2$ and $d = 3$), Iris dataset ($\gamma = 1$ and $d = 2$).

Running Time (*secs*)								
kernels	linear	rbf	poly	sigmoid	laplace	rbfpoly	linrbf	eta kernel
Moon	**0.11**	0.18	0.14	0.04	0.14	0.19	0.26	0.25
Circle	0.29	0.08	0.13	**0.08**	0.13	0.19	0.15	0.26
Classification	0.29	0.19	0.18	**0.22**	0.19	0.25	1.16	0.51
Iris	**0.002**	0.003	0.005	0.02	0.002	0.003	0.003	0.05

Table 6: Running time(in secs) for Moon dataset ($\gamma = 5$ and $d = 5$), Circle dataset ($\gamma = 5$ and $d = 7$), Classification dataset ($\gamma = 2$ and $d = 3$), Iris dataset ($\gamma = 1$ and $d = 2$).

4 CONCLUSION

In this paper we presented kmeans and it kernel versions using well known kernels. We also introduced the multikernel for kernel kmeans algorithm. We experimented on variety of datasets including *moon, circle, classification and iris datasets*. We show with the use of an evaluation metric (**Rand index**), that for non-linearly seperable data, kernel kmeans outperforms classic kmeans. for linearly seperable data (iris dataset), classic kmeans is sufficient for clustering.

References

[1] Laurence Morissette and Sylvain Chartier. The k-means clustering technique: General considerations and implementation in Mathematica. *Tutorials in Quantitative Methods for Psychology.* 9(1): pp. 15-24.

[2] Inderjit S. Dhillon, Yuqiang Guan, Brian Kulis. Kernel k-means, Spectral Clustering and Normalized Cuts. *[Information Search and Retrieval.* 9(1): pp. 15-24.

[3] Mehmet Gönen, Ethem Alpaydin. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research*, 12:2211-2268, 2011.