Kernel k-MACE: Hypercube Unsupervised Clustering Method

by

Faizan Ur Rahman

Bachelor of Applied Science, University of Toronto, 2015

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2017

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A THESIS

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Kernel k-MACE: Hypercube Unsupervised Clustering Method

Master of Applied Science 2017

Faizan Ur Rahman

Electrical and Computer Engineering

Ryerson University

Abstract

Transforming data to feature space using a kernel function can result in better expression of its features, resulting in better separability for some datasets. The parameters of the kernel function govern the structure of data in feature space and need to be optimized simultaneously while also estimating the number of clusters in a dataset. The proposed method denoted by kernel k-Minimum Average Central Error (kernel k-MACE), estimates the number of clusters in a dataset while simultaneously clustering the dataset in feature space by finding the optimum value of the Gaussian kernel parameter σ_k. A cluster initialization technique has also been proposed based on an existing method for k-means clustering. Simulations show that for self-generated datasets with Gaussian clusters having 10% - 50% overlap and for real benchmark datasets, the proposed method outperforms multiple state-of-the-art unsupervised clustering methods including k-MACE, the clustering scheme that inspired kernel k-MACE.

Acknowledgements

First of all I would like to thank my supervisor Dr. Soosan Beheshti for her support, encouragement and sincere advice throughout my MASc studies. For this, I am truly grateful.

I would also like to thank the members of my thesis committee: Dr. Ling Guan, Dr. Ebrahim Bagheri and Dr. Sri Krishnan for their valuable comments and recommendations.

I would also like to thank my colleague at Signal and Information Processing Lab, Edward Nidoy for his help and support as well as the technical discussions which were very beneficial for this research.

I would also like to thank my family for their love and support and hope that the completion of this milestone will make them happy.

Contents

Declaration ii

Abstract iii

Acknowledgements iv

List of Tables vii

List of Figures viii

List of Appendices ix

List of Abbreviations xiv

1 Introduction **1**

2 Background **5**

2.1 Kernel Methods 5

2.1.1 Kernel Basics 5

2.1.2 Calculation of Kernel Feature Map 8

2.1.3 Visualizing the Kernel Feature Map 9

2.1.4 Kernel k-means 9

2.2 k-MACE Algorithm 11

2.3 Index Validation Methods for Cluster Evaluation 12

2.3.1 Gap Index 12

2.3.2 Calinski-Harabasz Index 13
2.3.3 Davies-Bouldin Index 13
2.3.4 Silhouette Index 14

3 Kernel k-MACE Clustering **16**

3.1 Study and Analysis of Gaussian Kernel Functions 17
3.1.1 Gaussian Kernel Function 17
3.1.2 Examining Gaussian Kernel and σ_k 18
3.2 Kernel k-MACE Algorithm 23
3.2.1 Preliminaries and Notations 23
3.2.2 Initial Data Assignment and Clustering for Kernel k-MACE . . . 24
3.2.3 ACE and Data Error in Kernel k-MACE 26
3.2.4 Estimating $\overline{Z_{s_m}}$ and $\hat{m}$ 28
3.2.5 Choosing the Gaussian Kernel Parameter σ_k 29
3.2.6 Time Complexity of Kernel k-MACE 31

4 Simulations and Results **33**

4.1 Proposed Cluster Initialization Vs Random Cluster Initialization for Kernel k-means 33
4.2 Synthetic Datasets 34
4.3 Real Datasets 43
4.4 Normality Tests for Kernel k-MACE Evaluation 47

5 Conclusion and Future Work **49**

List of Tables

1 Table of symbols used for the formulation of k-MACE part 1 [29] xv

2 Table of symbols used for the formulation of k-MACE part 2 [29] xvi

3 Table of symbols used for the formulation of kernel k-MACE xvii

2.1 Common kernel functions 7

4.1 Clustering results for synthetic datasets S1 - S6 41

4.2 Clustering results for synthetic datasets S10 - S15 42

4.3 Real Datasets 43

4.4 Clustering results for real datasets 46

List of Figures

3.1 Surface in input space [31] 17

3.2 Gaussian kernel feature space representation of the surface in Figure 3.1 for different values of σ_k [31] 18

3.3 Plot of $\phi(S1)$ for different values of σ_k 20

3.4 Top view of $\phi(S1)$ for different values of σ_k 21

3.5 Plot of $\phi(S1)$ for $\sigma_k = 10^9$ and $\sigma_k = 10^{16}$ 21

3.6 Top view of $\phi(S4)$ for different values of σ_k 22

3.7 Z_{s_m} for dataset S1 27

3.8 Y_{s_m} for dataset S1 28

3.9 Plots showing $\overline{Z_{s_m}}$ for different values of σ_k from different angles 30

4.1 Proposed cluster initialization vs random cluster initialization 34

4.2 Datasets S1 - S6 36

4.3 Final clustering results for datasets S4, S5 and S6 using kernel k-MACE and k-MACE 38

List of Appendices

Appendix **52**

1 k-MACE **52**

List of Abbreviations

CNC ..... Correct Number of Cluster

SVD ..... Singular Value Decomposition

PCA ..... Principal Component Analysis

ACE ..... Average Central Error

NVI ..... Normalized Variation Index

ARI ..... Adjusted Random Index

KPCA ..... Kernel Principal Component Analysis

DBSCAN .... Density Based Spatial Clustering of Applications with Noise

Table 1: Table of symbols used for the formulation of k-MACE part 1 [29]

Symbol	Description
x^N	Dataset to be clustered
x_i	The i^{th} element of dataset x^N. Sampled from random vector X
N	Number of data samples in dataset
d	Dimension of data / Number of features
m	Number of clusters
$\hat{m}$	Estimated number of clusters
$\overline{m}$	Correct number of clusters
C_{mj}	j^{th} cluster in m-clustering step
$\overline{c}_{x_{mj}}$	True center of all the elements in cluster C_{mj}
$\hat{c}_{mj}$	Center of cluster C_{mj}
n_{mj}	Number of elements in cluster C_{mj}
x_{mj}	All the elements of cluster C_{mj}
x^i_{mj}	i^{th} element of cluster C_{mj}

Table 2: Table of symbols used for the formulation of k-MACE part 2 [29]

Symbol	Description
$\hat{W}_{x^i_{mj}}$	Independent random vector that describes variation of x^i_{mj} around its true center
$\overline{\wedge}_{x^i_{mj}}$	Covariance matrix of $\hat{W}_{x^i_{mj}}$
$Z_{s_{mj}}$	Central error of cluster C_{mj}
Z_{s_m}	Average central error of m-clustering step
$\overline{Z_{s_m}}$, $\underline{Z_{s_m}}$	Upper bound and lower bound of Z_{s_m}, respectively
$Y_{s_{mj}}$	Data error of cluster C_{mj}
Y_{s_m}	Average data error of m-clustering step

Table 3: Table of symbols used for the formulation of kernel k-MACE

Symbol	Description
ϕ_i	The i^{th} element of dataset ϕ^N. Sampled from random vector Φ
$\overline{c}_\Phi$	Center of random variable Φ
$\overline{W}_\Phi$	A dependent random vector that is describes variation of Φ around its center. $\overline{W}_\Phi$ is a zero mean Gaussian distribution
$C_{\phi_{mj}}$	j^{th} cluster in m-clustering step in feature space
$\overline{c}_{\phi_{mj}}$	True center of all the elements in cluster $C_{\phi_{mj}}$
$\hat{c}_{\phi_{mj}}$	Center of cluster $C_{\phi_{mj}}$
$n_{\phi_{mj}}$	Number of elements in cluster $C_{\phi_{mj}}$
ϕ_{mj}	All the elements of cluster $C_{\phi_{mj}}$
K	Kernel function
G	Kernelized distance matrix
σ_k	Gaussian kernel function parameter

Chapter 1

Introduction

Clustering is a branch of unsupervised learning and is defined as the process of organizing data samples into groups, such that a cluster contains samples which are similar to each other and are dissimilar to samples belonging to other clusters. Clustering is a widely researched field of machine learning and has applications in engineering, statistics, bioinformatics and finance etc. In computer science clustering algorithms are the backbone of search engines [1]. In market analysis, clustering is used for market segmentation which can be very helpful in identifying the target demographic of a product and in turn help with advertising [2]. In academics, class performance can be better represented using clustering which can be used as a reliable metric for evaluation of a professor's teaching methodology [3].

Clustering algorithms can be grouped into two main classes, hierarchical and partition based clustering algorithms [4]. Partition based clustering methods group data into non-overlapping clusters by iterative reassignment of data samples between clusters by minimizing a cost function. Hierarchical clustering methods build a hierarchy of clusters (dendrogram) and consist of an agglomerative or a divisive approach. In an agglomerative

approach, each data sample starts from its own cluster and gradually clusters are merged as we ascend the dendrogram, whereas in a divisive approach all of the data samples start in one cluster and clusters are divided as we descend the dendrogram. The splitting location of the dendrogram provides us with the clustering solution. The assumption regarding the structure of data being clustered is very important and governs the type of clustering algorithm that should be used to cluster it. Partitional clustering methods are generally used for clustering data with clusters having uniform or Gaussian distributions whereas data consisting of arbitrary shaped clusters is clustered using hierarchy based clustering algorithms.

In Unsupervised machine learning applications, kernel functions are used for better data representation for pattern recognition [5][6]. The resulting data is then used for tasks such as dimensionality reduction for feature selection and clustering [7]. Kernel functions have been widely used for clustering data and the kernel k-means algorithm is an implementation of k-means in a high dimensional space [8]. Kernel k-means is a partition based clustering algorithm that transforms data in input space to a higher dimensional (feature space) using a kernel function and then performs k-means clustering in feature space. Kernel based clustering algorithms are generally used for clustering non-linearly separable data. However they can also be used to better separate linearly separable data by transforming it to feature space.

Kernel k-means has a high time complexity for very large datasets as the algorithm computes the gram matrix for the input dataset, which contains the distance between each data sample in feature space. Algorithms have been proposed to make kernel k-means faster with little loss in clustering quality [9][10]. The single pass kernel k-means clustering method proposes a method that traverses the dataset only once to compute the clustering result and significantly reduces the computing time of the kernel k-means

algorithm [11]. Other approaches to reduce the computing time of Kernel k-means include distributing the workload on multiple machines as proposed in [12].

Kernel k-means can also be used for clustering datasets that consist of clusters from different distributions. The method proposed by [13] uses a unique kernel for each localized distribution of data resulting in better clustering results. Kernel k-means can also be used to find overlapping clusters along with non-linearly separable clusters as proposed in [14]. It should be noted that all the clustering methods stated above are not completely unsupervised and require the number of clusters as an input.

Knowledge of the correct number of clusters (CNC) in a dataset is required as an input by most clustering algorithms and an incorrect value of the number of clusters can lead to incorrect clustering results. As CNC is not available in real life which is why many clustering techniques focus on correctly estimating the CNC for clustering data and these clustering techniques are called fully unsupervised clustering algorithms [15]. Kernel based clustering algorithms require the kernel parameters as an input in addition to the number of clusters.

Fully unsupervised methods include kernel methods such as bee colony optimization [16]. This method uses an evolutionary algorithm with kernel functions to correctly estimate the number of clusters and assign data samples to these clusters. Another fully unsupervised algorithm that uses Kernel Principal Component Analysis (KPCA) to create rough clusters before using swarm intelligence to obtain the final clustering result is proposed in [17]. Other methods proposed in [18] and [19] also obtain the CNC as well as the correct clustering result using kernel functions. None of these methods estimate the value of the kernel function parameter and use trial and error method to find the parameter value that produces the best results.

The most common fully unsupervised clustering methods still rely on using internal

validation indices to estimate the number of clusters in a dataset. A validation index evaluates the clustering result based on the number of clusters chosen. The most commonly used validation indices are Gap [20], Calinski-Harabasz [21], Davies-Bouldin [22] and Silhouette index [23]. These validation indices work alongside a clustering algorithm which requires the number of clusters as an input [24]. The clustering algorithm clusters data for each k (k is the number of clusters) from a range of values $k = [k_{min}, k_{min}+1, ..., k_{max}]$. The validation index evaluates the clustering result for each k and chooses the value of k which optimizes the index. If a kernel based clustering algorithm is used alongside validation indices, the kernel parameter can be estimated from a provided range by choosing the optimum value of the validation index which would correspond to both the estimated number of clusters as well as the estimated parameter value.

Our Objectives: The focus of this thesis is to demonstrate the use of kernel functions to create a fully unsupervised clustering algorithm based on the k-MACE clustering scheme. We also aim to improve the clustering results of k-MACE for datasets containing Gaussian clusters with various degrees of overlap.

Thesis Outline: This thesis is organized as follows: In Chapter 2, a background of kernel functions and kernel methods is provided. The fully unsupervised clustering scheme k-MACE, which inspired kernel k-MACE is also provided. In Chapter 3, Kernel k-Minimizing Average Central Error (kernel k-MACE) for estimating the number of clusters and the Gaussian kernel parameter for a dataset is proposed. Simulations and results are discussed in Chapter 4. Conclusion and future works are provided in Chapter 5.

Chapter 2

Background

In this Chapter, we review kernel functions and their applications for clustering data.

In Section 2.1, we discuss the basics of kernels including the characteristics of a kernel function, the kernel trick and commonly used kernel functions. The procedure for calculating and visualizing the kernel feature map has also been discussed. The most common kernel based clustering technique, kernel k-means has been discussed.

Section 2.2 discusses the k-MACE algorithm which is the motivation behind the kernel k-MACE algorithm. The k-MACE algorithm performs signal denoising through best basis selection and estimation of an unobservable error.

Section 2.3 discusses some common external validation indices.

2.1 Kernel Methods

2.1.1 Kernel Basics

Conventional partitional clustering methods including k-means, Fuzzy C-means and K-medoids are efficient in clustering data in the input space [25][26]. The objective of k-

means is to minimize a cost function by iteratively calculating distance between samples in a dataset. Kernel functions are used to transform the data into a higher dimensional space (called feature space F) and obtain the distance matrix of the dataset in feature space F [27]. Distance matrix of a dataset contains the distance between data samples in the dataset using a distance measure such as the Euclidean distance.

Transformation of a dataset into a higher dimensional space results in a better expression of features in the dataset making it easier to cluster using a clustering algorithm. The biggest advantage of using a kernel function is that it calculates the distance between data samples in F without requiring any knowledge of the transformation function ϕ (also known as feature map). Equation (2.1) defines a kernel function K in terms of ϕ.

$$K(X_i, X_j) = \phi(X_i)^T \phi(X_j) \tag{2.1}$$

The distance between two data samples X_i and X_j in F is defined by Equation (2.2). The distance Equation (2.2) is simplified using the kernel trick by computing the dot product in F using the kernel function K resulting in no need for the knowledge of ϕ, where $\phi = X \rightarrow F$.

$$\begin{aligned} ||\phi(X_i) - \phi(X_j)||^2 &= \phi^2(X_i) - 2\phi(X_i)\phi(X_j) + \phi^2(X_j) \\ &= K(X_i, X_i) + K(X_j, X_j) - 2K(X_i, X_j) \end{aligned} \tag{2.2}$$

Commonly used kernel functions are given in Table 2.1.

Linear kernel	$K(X_i, X_j) = {X_i}^T X_j$
Polynomial kernel	$K(X_i, X_j) = ({X_i}^T X_j + \gamma)^\delta$
Gaussian kernel	$K(X_i, X_j) = \exp(\frac{-\|X_i - X_j\|^2}{2\sigma^2})$
Sigmoid kernel	$K(X_i, X_j) = \tanh(\gamma({X_i}^T X_j) + \theta)$

Table 2.1: Common kernel functions

For K to be a kernel function, it must satisfy the following conditions:

- There must exist a feature space F for which K defines a dot product.
- G must be a symmetric matrix. Where G is the kernelized distance matrix obtained using K for the corresponding dataset.
- G must be a positive semi-definite matrix (all eigenvalues should be positive).
- G must be an $n \times n$ matrix of pairwise distance between data samples in F for a dataset X, where X contains n data samples.

Let $X \in R^{n \times d}$ be the input dataset with n samples and d dimensions. The kernelized distance matrix G for the dataset is defined as $G \in R^{n \times n}$. The distance between the data samples X_i and X_j in F is defined by Equation (2.2). $\phi(X_i)$ and $\phi(X_j)$ are feature mappings of points X_i and X_j in F.

The feature map for a kernel function can be derived mathematically. The feature map for Polynomial kernel with parameters $\gamma = 0$ and $\delta = 2$ is derived in Equation (2.3), where X_i and X_j are 2D data samples. The kernel function transforms the dataset from

2D to a 3D space.

$$
\begin{aligned}
K(\begin{pmatrix} X_{i1} \\ X_{i2} \end{pmatrix}, \begin{pmatrix} X_{j1} \\ X_{j2} \end{pmatrix}) &= (X_{i1}X_{j1} + X_{i2}X_{j2})^2 \\
&= (X_{i1}X_{j1})^2 + (X_{i2}X_{j2})^2 + 2X_{i1}X_{j1}X_{i2}X_{j2} \\
&= \left(X_{i1}^2, \sqrt{2}X_{i1}X_{i2}, X_{i2}^2\right)^T \begin{pmatrix} X_{j1}^2 \\ \sqrt{2}X_{j1}X_{j2} \\ X_{j2}^2 \end{pmatrix}
\end{aligned}
\tag{2.3}
$$

2.1.2 Calculation of Kernel Feature Map

It is difficult and time consuming to derive the feature map of a given kernel function analytically and a general method is needed to obtain the feature map for any kernel function which is given below.

1. Obtain the kernelized distance matrix G for a dataset using a kernel function K.

2. Decompose matrix G using Singular Value Decomposition (SVD) which is a technique used to decompose a positive semi-definite matrix into its eigenvalues and eigenvectors. SVD is used to decompose G into three matrices U, D and U^T given by Equation (2.4).

   - Columns of matrix U contain the eigenvectors of G.
   - D is a diagonal matrix containing eigenvalues of matrix G on its diagonal.
   - U^T is the transpose of matrix U and $UU^T = I$, where I denotes the identity matrix.

$$G = UDU^T \tag{2.4}$$

3. The feature map of the complete dataset is denoted by $\phi(X) \in R^{n \times n}$ and is obtained using the formula $\phi(X) = (\sqrt{D}U^T)^T$. Each row of $\phi(X)$ denotes the value of each sample in F with each column denoting the dimension [28].

2.1.3 Visualizing the Kernel Feature Map

Feature maps depend completely on the kernel functions they are created by. Kernel feature map is used to visualize the data in feature space however, the first two or three principal components cannot completely show the structure of data in feature space. The dimensions of data samples in feature space (using the method stated above) are equal to the number of data samples in the dataset. Therefore the feature space for Polynomial kernel of degree 2 would have N dimensions rather than 3 dimensions as in Equation (2.3). To visualize a dataset with N dimensions in kernel space, we use SVD to obtain the feature map ϕ of the dataset and then use PCA. Kernel PCA can also be used to obtain the principal components of ϕ directly using the dataset X to visualize it using 3D plots.

2.1.4 Kernel k-means

A clustering method is needed to cluster a dataset after transforming it to feature space using a kernel function K. Kernel k-means is a clustering algorithm similar to the k-means clustering algorithm which uses kernel functions to calculate the distance between data samples in feature space [8]. Kernel k-means clustering algorithm is used to cluster data that is usually non-linearly separable in input space. Kernel k-means only returns

the cluster memberships of all the data samples as compared to both cluster memberships and cluster centers for each cluster in k-means.

Algorithm 1 Kernel k-means Algorithm

Require: Provide a clustering solution.
Input: Data set $x^N = [x_1, x_2, ..., x_N]$, number of clusters m and parameters of the kernel function being used
Output: The clustering solution $[C_1, C_2, .., C_k]$

```
Initialize each sample in the dataset to a random cluster
while new[C_1, C_2, .., C_k] ≠ old[C_1, C_2, .., C_k] do
    for each data sample x_i, i = 1, ..., N do
        for each cluster C_j, j = 1, .., k do
            Calculate number of elements in cluster |C_j|
            Calculate G_k(C_j) using (2.7)
            Calculate F_k(X_i, C_j) using (2.6)
            Find ||φ(X_i) − μ_j||^2 using (2.5)
        end for
        Assign x_i to its nearest cluster C_j using C(x_i) = argmin(||φ(X_i) − μ_j||^2)
    end for
end while
```

There is no need to explicitly compute the centroid μ_j for each cluster as the kernel trick simplifies the cost function in Equation (2.5). The cost function and F_k and G_k are defined in Equations (2.5), (2.6) and (2.7) respectively.

$$\begin{aligned} \|\phi(X_i) - \mu_j\|^2 &= \|\phi(X_i) - \sum_{X_l \in C_j} \frac{\phi(X_l)}{|C_j|}\|^2 \\ &= \phi(X_i)\phi(X_i) - F_k(X_i, C_j) + G_k(C_j) \end{aligned} \tag{2.5}$$

$$F_k(X_i, C_j) = -\frac{2}{|C_j|} \sum_{X_l \in C_j} \phi(X_l)\phi(X_i) \tag{2.6}$$

$$G_k(C_j) = \frac{1}{|C_j|^2} \sum_{X_l \in C_j} \sum_{X_s \in C_j} \phi(X_l)\phi(X_s) \tag{2.7}$$

2.2 k-MACE Algorithm

k-MACE is a fully unsupervised data clustering algorithm that clusters a dataset while also estimating the Correct Number of Clusters (CNC) in the dataset [29]. k-MACE initially clusters a dataset using the k-means algorithm. Statistical calculations shown in Appendix 1 are performed on the clustering result to obtain the cluster compactness and eventually estimate the number of clusters. The final clustering result is obtained using the k-means clustering algorithm. The equations for k-MACE algorithm have been provided in Appendix 1.

Algorithm 2 k-MACE Algorithm

Require: Estimate the number of clusters $\hat{m}$ and provide a clustering solution.
Input: Data set $x^N = [x_1, x_2, ..., x_N]$, range of m, $[m_{min}, m_{max}]$
Output: Estimated number of clusters $\hat{m}$, and the clustering solution $[\hat{C}_1, \hat{C}_2, .., \hat{C}_{\hat{m}}]$

1: **for** $(m = m_{min}; m \leq m_{max}; m_{++})$ **do**
2: $[C_{m1}, C_{m2}, .., C_{mj}] = kmeans(x, m)$
3: **for** each cluster C_{mj} $j = 1, .., m$ **do**
4: Solve cluster compactness y_{smj} of cluster C_{mj} using (1.4)
5: **end for**
6: Solve for total cluster compactness y_{sm} using (1.3)
7: **end for**
8: Solve for $\hat{\wedge}_{x_i}$ from clustering solution $[\hat{C}^0_1, .., \hat{C}^0_{\hat{m}^0}]$
9: Use $\hat{\wedge}_{x_i}$ and set $\alpha = \sqrt[3]{n_{mj}}$ to solve for the upperbound of $\|A_{mj}\bar{c}_{mj}\|^2_F$
10: Use $\hat{\wedge}_{x_i}$ to solve for $E[Z_{s_{mj}}]$ using (1.12), (1.13) and $Var[Z_{s_{mj}}]$ using (1.14)
11: Set $\beta_N = N$. The upperbound of Z_{s_m}, $\overline{Z_{s_m}}$, can then be found using (1.17).
12: $\hat{m} = arg\ min_m(\overline{Z_{s_m}})$
13: $[\hat{C}_1, \hat{C}_2, .., \hat{C}_{\hat{m}}] = kmeans(x, \hat{m})$

k-MACE results in correct estimation of number of clusters as well as accurate clustering of synthetic datasets which have clusters resulting from Gaussian distributions.

2.3 Index Validation Methods for Cluster Evaluation

Index validation methods are commonly used for evaluating the performance of a clustering algorithm which makes them very attractive for estimating the CNC in a dataset. External clustering validation indices such as ARI and NVI require the original labels of the data samples to be known to evaluate the clustering result. However internal validation indices such as Gap, Calinski-Harabasz, Davies-Bouldin and Silhouette use the clustering result of a dataset to evaluate the clustering performance. These indices can be used with any partitional or hierarchical clustering algorithm.

2.3.1 Gap Index

For a given dataset $x^N = [x_1, x_2, ..., x_N] \in R^{N \times d}$ where each sample has d dimensions, the distance between two data samples in denoted by $d_{i,j} = \sum(x_i - x_j)^2$. The clustering result after clustering the data into k clusters is denoted by $[C_1, C_2, ..., C_k]$. The sum of pairwise distances between all data samples belonging to cluster s is given by Equation (2.8).

$$D_s = \sum_{i,j \in C_k} d_{i,j} \tag{2.8}$$

The within cluster sum of squares around cluster mean for all clusters is given by Equation (2.9) where $|C_s|$ denotes the number of data samples in cluster s. The Gap index is then given by Equation (2.10) where E_n denotes the expected value of a sample of size n from the reference data distribution. The estimated number of clusters $\hat{m}$ is equal to $arg\max_m(GAP)$ [20].

$$W_k = \sum_{s=1}^{k} \frac{1}{|C_s|} D_s \tag{2.9}$$

$$GAP(k) = E_n(log(W_k)) - log(W_k) \tag{2.10}$$

2.3.2 Calinski-Harabasz Index

Calinski-Harabasz index is given by Equation (2.11). BGSS is the between cluster dispersion defined by Equation (2.12) where c_j denotes the center of cluster j and μ_x denotes the mean of the dataset x^N. WGSS is the within group scatter defined by Equation (2.13). The estimated number of clusters $\hat{m}$ is equal to $arg\max_m(CH)$ [21].

$$CH(k) = \frac{BGSS(N-k)}{WGSS(k-1)} \tag{2.11}$$

$$BGSS = \sum_{j=1}^{k} |Cj| \|c_j - \mu_x\| \tag{2.12}$$

$$WGSS = \sum_{j=1}^{k} \sum_{i=1}^{|Cj|} \|x_j^i - c_j\| \tag{2.13}$$

2.3.3 Davies-Bouldin Index

For a given clustering result, δ_j denotes the mean distance of data samples in the cluster j to its center c_j given by Equation (2.14). The difference between centers of clusters j and k is given by Equation (2.15). The Davies-Bouldin index is then given by Equation (2.16). The estimated number of clusters $\hat{m}$ is equal to $arg\min_m(DB)$ [22].

$$\delta_j = \frac{1}{|C_j|} \sum_{i \in j} \|x_j^i - c_j\| \tag{2.14}$$

$$\Delta_{j,k} = \|c_j - c_k\| \tag{2.15}$$

$$DB_k = \frac{1}{k}\sum_{j=1}^{k} max_{j \neq j'} \frac{\delta_j + \delta_{j'}}{\Delta_{j,j'}} \tag{2.16}$$

2.3.4 Silhouette Index

For a given clustering result, we calculate the average distance between a data sample i in cluster j and every other data sample in cluster j. Call this distance a_i. For a data sample i not belonging to a cluster j, we calculate the average distance of this data sample to each data sample in the cluster j and find the minimum of distance with respect to all clusters. Call this value b_i. The Silhouette coefficient is then given by Equation (2.17)

$$S(i) = \frac{b_i - a_i}{max(a_i, b_i)} \tag{2.17}$$

The value of the Silhouette coefficient varies between 1 and -1 and the results closer to 1 denote a better clustering result. The overall Silhouette coefficient S for a dataset is obtained by averaging $S(i)$ for the whole dataset. The estimated number of clusters $\hat{m}$ is equal to $arg\max_m(S)$ [23].

Chapter 3

Kernel k-MACE Clustering

Unsupervised clustering methods that use kernel functions to cluster data in feature space require two input parameters, the number of clusters in the dataset and the optimum value of the kernel function parameter. Kernel based unsupervised clustering methods, which estimate the number of clusters in a dataset [18] and [19] use the best possible value of the kernel parameter by using trial and error method. The proposed method estimates the correct number of clusters while also estimating the value of the Gaussian kernel parameter σ_k that corresponds to the best clustering result for most datasets.

Section 3.1 discusses the Gaussian kernel function and the effect of changing the Gaussian kernel parameter on data in feature space. Section 3.2 presents the proposed method and its time complexity.

3.1 Study and Analysis of Gaussian Kernel Functions

3.1.1 Gaussian Kernel Function

The Gaussian kernel function is stated in Table 2.1. Data samples transformed using Gaussian kernel lie on a hypersphere in feature space which is centered about the origin and has a radius of 1 [30]. These data samples have the following properties.

- All the data samples which are N dimensional vectors, are linearly independent i.e. no data sample can be represented by a linear combination of other data samples.
- The feature space of a Gaussian kernel has infinite dimensions.

Even though the feature space F for the Gaussian kernel is infinite dimensional, for a dataset containing finite samples N, the feature map ϕ can be approximated using an N dimensional feature map. The kernel feature map of a surface in input space in Figure 3.1 has been shown in Figure 3.2 for different values of the parameter σ_k of the Gaussian kernel. It should be noted that $\gamma = \frac{1}{2\sigma_k^2}$.

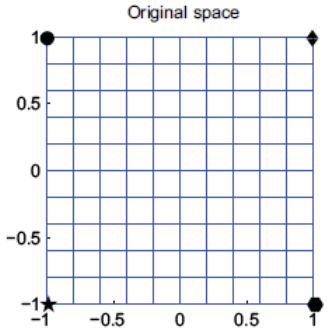


Figure 3.1: Surface in input space [31]

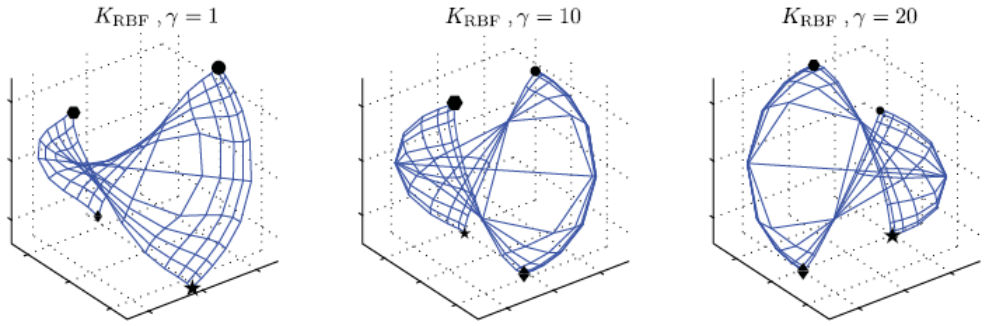


Figure 3.2: Gaussian kernel feature space representation of the surface in Figure 3.1 for different values of σ_k [31]

An explicit definition of the Gaussian kernel feature map ϕ_{Gauss} using Taylor series expansion is given in Equation (3.1) where σ_k is the Gaussian kernel parameter.

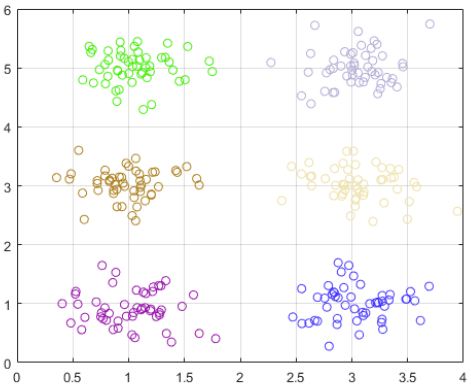
$$\phi_{Gauss}(x) = e^{-\frac{x^2}{2\sigma_k^2}}[1, \sqrt{\frac{1}{1!\sigma_k^2}}x, \sqrt{\frac{1}{2!\sigma_k^4}}x^2, \sqrt{\frac{1}{3!\sigma_k^6}}x^3, ...]^T \qquad where\ x \in R^{1\times d} \tag{3.1}$$

3.1.2 Examining Gaussian Kernel and σ_k

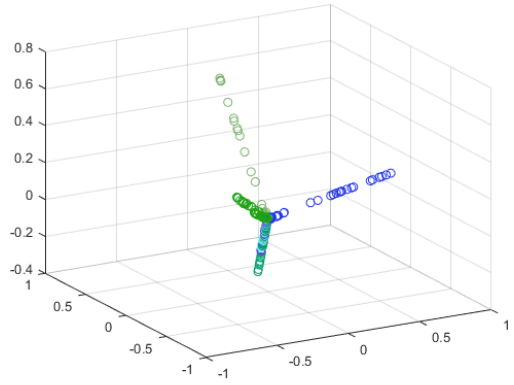
The dataset that needs to be clustered should be transformed into feature space such that the clusters are easily separable. The feature map depends solely on the kernel function and its parameters used. The parameter used with Gaussian kernel is called σ_k for the entirety of this thesis. Consider the dataset S1 in Figure 3.3a. The dataset consists of 6 clusters which can be easily separated using a clustering algorithm however to demonstrate the effect of σ_k on the kernel feature map we have used SVD and PCA to plot the first 3 principal components of the kernel feature map for different values of σ_k. The plots of the first three principal components $\phi(S1)$ for values of $\sigma_k = 0.01$, 0.1, 1, 10 and 100 are shown in Figures 3.3b - 3.3f.

The top view of Figures 3.3d - 3.3f is shown in Figures 3.4a - 3.4c. The optimum

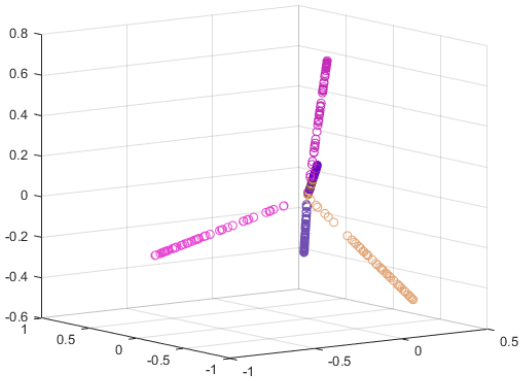
value of σ_k has to be selected to obtain the best separation of clusters within a dataset S1. From Figures 3.3 and 3.4 we can see that σ_k values less than 1 will not generate a good separation of clusters in feature space. σ_k values of 1 and larger produce results which show well separated clusters having a familiar looking profile as in the input space. The profile of data is not completely visible in Figures 3.3d - 3.3f as PCA has been applied to view $\phi(S1)$ which has N dimensions equal to the number of samples in the original dataset. Very large values of σ_k such as 10^9 will still result in a good cluster separation in feature space while retaining the profile of the clusters as shown in Figure 3.5a. Increasing σ_k further will result in the loss of the separation at $\sigma_k = 10^{16}$ as shown in Figure 3.5b. The range of σ_k that results in the visible separation of clusters is different for different datasets. Figures 3.3 - 3.5 we can see that kernel transformation is not required to separate data and S1 can be easily separated in input space.



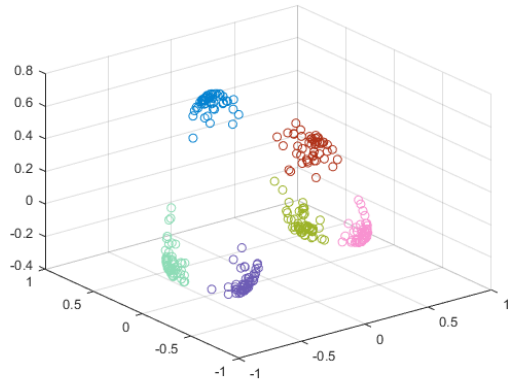
(a) Synthetic dataset S1



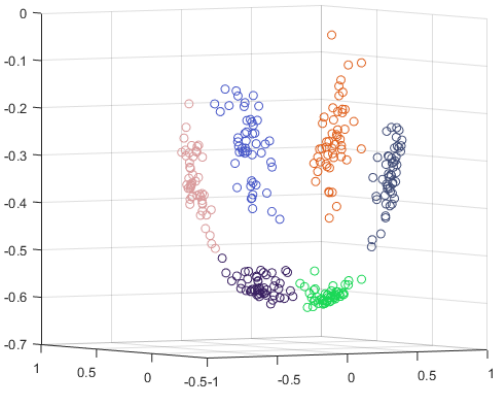
(b) $\phi(S1)$ for $\sigma_k = 0.01$



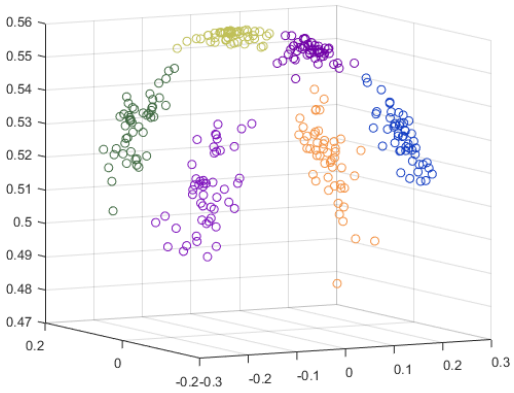
(c) $\phi(S1)$ for $\sigma_k = 0.1$



(d) $\phi(S1)$ for $\sigma_k = 1$

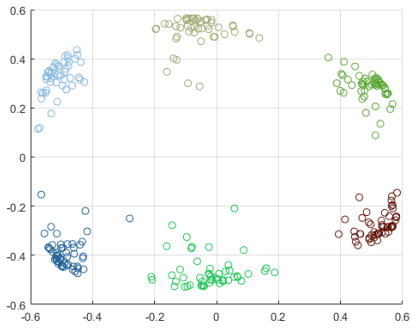


(e) $\phi(S1)$ for $\sigma_k = 10$

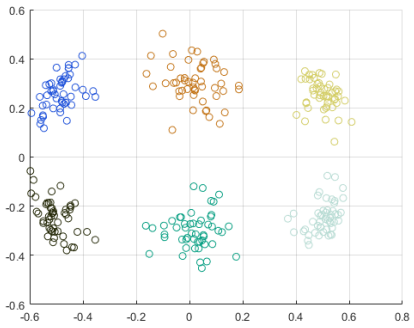


(f) $\phi(S1)$ for $\sigma_k = 100$

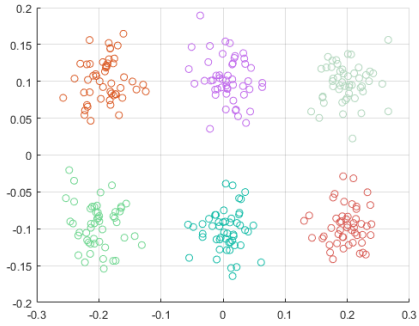
Figure 3.3: Plot of $\phi(S1)$ for different values of σ_k



(a) Top view of $\phi(S1)$ for $\sigma_k = 1$

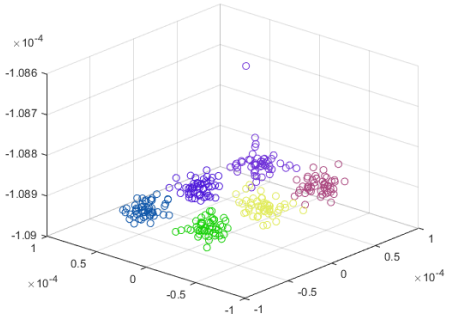


(b) Top view of $\phi(S1)$ for $\sigma_k = 10$

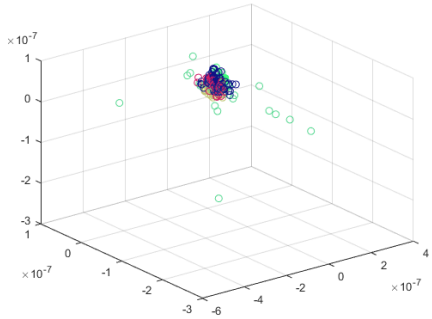


(c) Top view of $\phi(S1)$ for $\sigma_k = 100$

Figure 3.4: Top view of $\phi(S1)$ for different values of σ_k



(a) $\phi(S1)$ for $\sigma_k = 10^9$

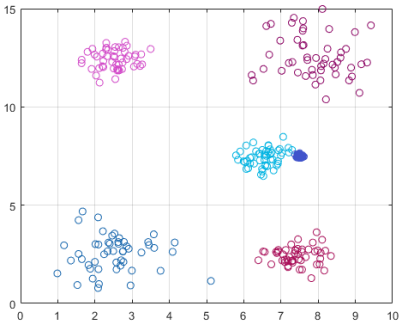


(b) $\phi(S1)$ for $\sigma_k = 10^{16}$

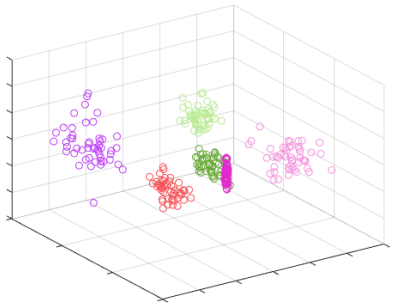
Figure 3.5: Plot of $\phi(S1)$ for $\sigma_k = 10^9$ and $\sigma_k = 10^{16}$

Now concider the dataset S4 in Figure 4.2d. This dataset consists of 2 clusters with

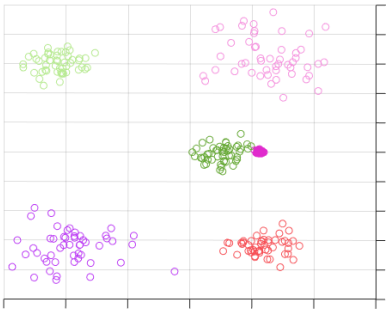
very different variances in close proximity of each other. The feature space representation of the dataset using Gaussian kernel and values of $\sigma_k = 50$ are shown in Figure 3.6b. The top view for the same plot is shown in Figure 3.6c. Figure 3.6b clearly shows the purple cluster (having a Gaussian profile) beside the dark green cluster in 3D. The dataset S4 should be successfully clustered using kernel methods as kernel functions obtain a better representation of data in feature space.



(a) Dataset S4



(b) $\phi(S4)$ for $\sigma_k = 50$



(c) Top view of $\phi(S4)$ for $\sigma_k = 50$

Figure 3.6: Top view of $\phi(S4)$ for different values of σ_k

3.2 Kernel k-MACE Algorithm

Kernel k-MACE is based on the k-MACE clustering scheme but in comparison, both clustering and the cluster evaluation of a dataset are done in the feature space F. To accomplish this the kernel k-means algorithm has been combined with the k-MACE algorithm to obtain kernel k-MACE algorithm. The clustering technique presented estimates the number of clusters present in a dataset and evaluates the best partition while simultaneously estimating the optimum value for Gaussian kernel parameter σ_k from a range of provided values.

3.2.1 Preliminaries and Notations

Tables 1 and 2 provide the list of symbols used for formulation of k-MACE. The corresponding equations are provided in Appendix 1. Table 3 provides the list of symbols used for the formulation of kernel k-MACE.

Given a dataset of length N in feature space, $\phi^N = [\phi_1, \phi_2, \ldots, \phi_N]^T$ where $\phi_i \in R^{1\times d}$. Each data sample ϕ_i is an array of length N, and each element of ϕ_i represents a feature. The data model is given by Equation (3.2). Each sample ϕ_i of Φ results from zero mean Gaussian noise (denoted by $\overline{W}_\Phi$) being added to its mean (denoted by $\bar{c}_\Phi$).

$$\Phi = \bar{c}_\Phi + \overline{W}_\Phi \tag{3.2}$$

The dataset consists of $\bar{m}$ clusters where each data sample only belongs to one cluster.

$$\phi^N = \overline{C}_1 \cup \overline{C}_2 \cup ... \cup \overline{C}_{\overline{m}} \tag{3.3}$$

Data samples belonging to each cluster C_j can be represented by Equation (3.4).

$$\begin{bmatrix} \phi_1 \\ \vdots \\ \phi_{n_j} \end{bmatrix} = \begin{bmatrix} \overline{c}_{\phi_j} \\ \vdots \\ \overline{c}_{\phi_j} \end{bmatrix} + \begin{bmatrix} \mathcal{N}(0, \overline{\Sigma}_j) \\ \vdots \\ \mathcal{N}(0, \overline{\Sigma}_j) \end{bmatrix} \tag{3.4}$$

3.2.2 Initial Data Assignment and Clustering for Kernel k-MACE

The input dataset needs to be clustered initially just as in k-MACE (Algorithm 2: step 2). k-MACE employs the k-means algorithm to cluster data in the input space while kernel k-MACE uses kernel k-means to cluster data in feature space.

Kernel k-means requires every sample in the dataset to be assigned to a cluster initially. The original kernel k-means algorithm assigns every data sample to a cluster randomly. An algorithm has been proposed to improve accuracy of clustering results produced by the k-means algorithm in [32]. The same analogy has been used to extend the algorithm to work with kernel k-means given in Algorithm 3. To our knowledge the algorithm proposed in [32] has not been used to work with kernel methods. The method in Algorithm 3 makes the kernel k-means algorithm converge faster.

Algorithm 3 Initial cluster assignment

Require: Assign each data sample in x to a cluster.
Input: Data set $x^N = [x_1, x_2, ..., x_N]$, number of clusters m
Output: Initial Clusters $[C_1, C_2, .., C_m]$

1: **while** $new[C_1, C_2, .., C_k] \neq old[C_1, C_2, .., C_k]$ **do**
2: Compute kernelized distance given by (2.2) between each data sample x_i and all other data samples in x^N.
3: Find closest pair of points in x^N and form a set $A_i (1 \leq i \leq m)$ containing the pair. Remove the pair of points from x^N.
4: Find a data sample in x^N that is closest to A_i. Remove the data sample from the x^N.
5: Repeat step 4 until the number of data points in A_i reaches $0.75 * (N/m)$.
6: **if** i < m **then**
7: i = i + 1 and repeat steps 2 - 5
8: **end if**
9: Find the mean of the data samples in A_i to obtain the initial centroids $[\hat{c}_1, \hat{c}_2, .., \hat{c}_m]$ for each set $A_i (1 \leq i \leq m)$.
10: Compute kernelized distance between each data sample and all the centroids obtained in step 9.
11: Find the closest centroid for each data sample and assign the to the cluster represented by the centroid.
12: Recalculate the centroids for each cluster.
13: Compute the distance of each data sample from the centroid of the nearest cluster.
14: **if** The distance $\geq$ the current cluster centroid distance **then**
15: The data sample stays in the cluster.
16: **else**
17: Compute kernelized distance between each data sample and all the centroids. Assign the data sample to the cluster with the nearest centroid.
18: **end if**
19: Recalculate the centroids for each cluster
20: Repeat steps 13 - 19 until convergence
21: **end while**

Kernel k-means (Algorithm 1 in Chapter 2) takes as inputs, the initial clustering assignments $[C_1, C_2, .., C_m]$ for dataset x^N, the number of clusters and the kernel function parameters. The initial clustering assignments are obtained using Algorithm 3.

3.2.3 ACE and Data Error in Kernel k-MACE

Average Central Error (ACE) and data error have been used in the k-MACE algorithm [29]. k-MACE defines ACE in Equations (1.1) and (1.2) and the data error in Equations (1.3) and (1.4) in Appendix 1 [29]. Average Central Error (ACE) is used for evaluating the clustering results from kernel k-MACE for the provided range of $m : [m_{min}, ..., m_{max}]$. The minimum value of ACE corresponds to the estimated number of clusters in a dataset. The true ACE for a dataset is not available but its upper and lowerbounds can be estimated using probabilistic measures. The minimum of the upperbound of ACE will correspond to the estimated number of clusters $\hat{m}$ in a dataset. ACE is denoted by Z_{s_m} and is defined as the average distance between the estimated cluster center and the true cluster center. Kernel k-MACE defines $Z_{s_{mj}}$ as Equation (3.5) where $||(*)||_F$ refers to the Frobenius norm. Z_{s_m} is then calculated using Equation (1.1). The difference exists because we want to obtain Z_{s_m} for a dataset in feature space. Figure 3.7 shows a plot of Z_{s_m} for dataset S1.

$$Z_{s_{mj}} = \left\| \overline{c}_{\phi_{mj}} - \hat{c}_{\phi_{mj}} \right\|_F^2 \tag{3.5}$$

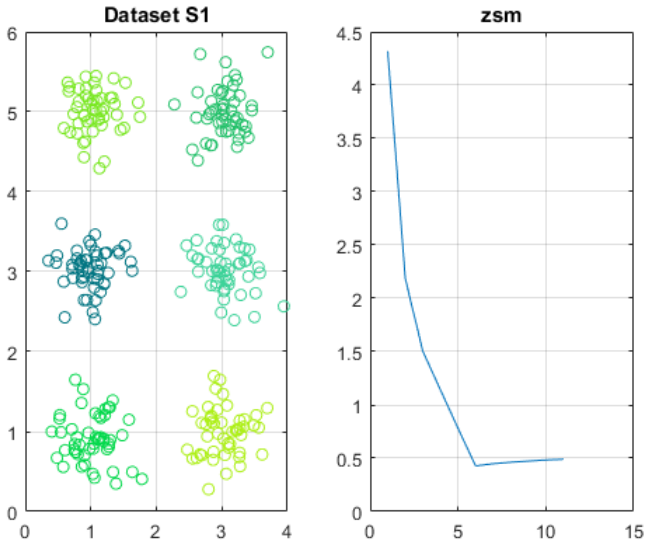


Figure 3.7: Z_{s_m} for dataset S1

$\overline{c}_{\phi_{mj}}$ is a matrix containing true cluster centers corresponding to each data sample in cluster $C_{\phi_{mj}}$ in feature space and $\hat{c}_{\phi_{mj}}$ is a matrix containing the estimated cluster centers corresponding to each data sample in cluster $C_{\phi_{mj}}$ in feature space from a clustering solution.

It should be kept in mind that the Z_{s_m} being talked about here is the true Z_{s_m} which is not known and Equations (1.1) and (3.5) are only used to define it. Our goal is to estimate the upper and lower bounds of the Z_{s_m}.

Data error is denoted by Y_{s_m} and is defined as the distance of data samples assigned to a cluster from the estimated cluster center otherwise known as cluster compactness defined in Equations (1.3) and (1.4) for k-MACE in Appendix 1. The data error is calculated for each clustering result from kernel k-means for the provided range of m and is further used to calculate $\overline{Z_{s_m}}$ which is the Z_{s_m} upperbound. Kernel k-MACE defines $Y_{s_{mj}}$ in Equation (3.6) where ϕ_{mj} is a data sample. Y_{s_m} is then calculated using Equation

(1.3). Figure 3.8 shows a plot of Y_{s_m} for dataset S1.

$$Y_{s_{mj}} = \left\| \hat{c}_{\phi_{mj}} - \phi_{mj} \right\|_F^2 \tag{3.6}$$

Note that both the data samples and the estimated cluster centers are in feature space F and are obtained by performing SVD on the kernelized distance matrix G.

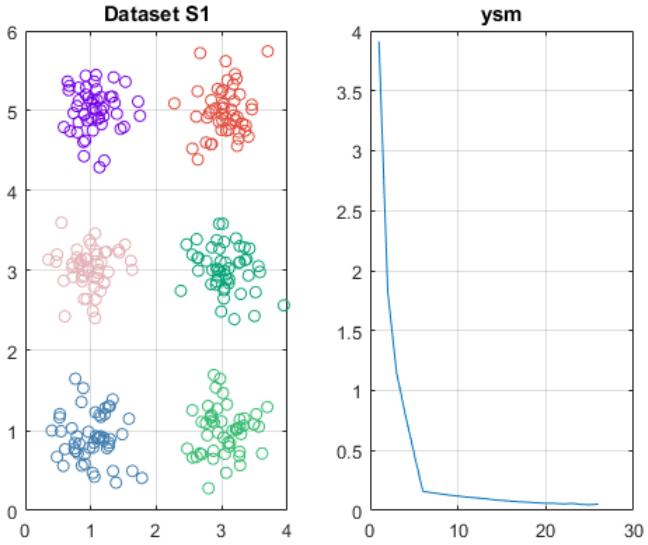


Figure 3.8: Y_{s_m} for dataset S1

3.2.4 Estimating $\overline{Z_{s_m}}$ and $\hat{m}$

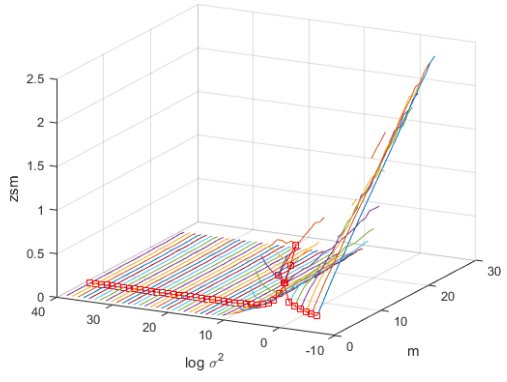
The probabilistic bounds of Z_{s_m} can be calculated using Equations (1.5) - (1.18) in Appendix 1. $\hat{m}$ is then calculated from $\overline{Z_{s_m}}$ using equation (3.7)

$$\hat{m} = arg \min_m (\overline{Z_{s_m}}) \tag{3.7}$$

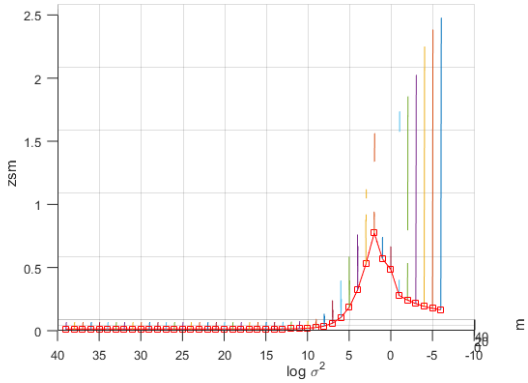
3.2.5 Choosing the Gaussian Kernel Parameter σ_k

Kernel based unsupervised clustering methods find the kernel parameter based on trail and error and dont have an automatic method of estimating it [16][17][18][19]. We obtain the clustering result for a range of values of σ_k. A plot of the $\overline{Z_{s_m}}$ for the S1 dataset (Figure 3.3a) for each value of σ_k is shown in Figure 3.9a. The side and top view of the Figure are shown in Figures 3.9b and 3.9c respectively. We propose a method to obtain the optimum value of σ_k that corresponds to the correct clustering result. The red squares denote the minimum value of $\overline{Z_{s_m}}$ i.e. $\hat{m}$ for each σ_k. To obtain the best σ_k, we obtain the gradient of the red curve shown in Figure 3.9a and for each σ_k, add the absolute value of the previous and the next gradient in Equation (3.8) after the peak. The value of σ_k which corresponds to the maximum value of this sum is chosen and the corresponding $\hat{m}$ and clustering result are chosen as the correct result. This corresponds to the 10th red square from the right which occurs at $\log\sigma_k^2 = 3$ and is visible in Figure 3.9b which corresponds to $\hat{m} = 6$ which is visible in Figure 3.9c.

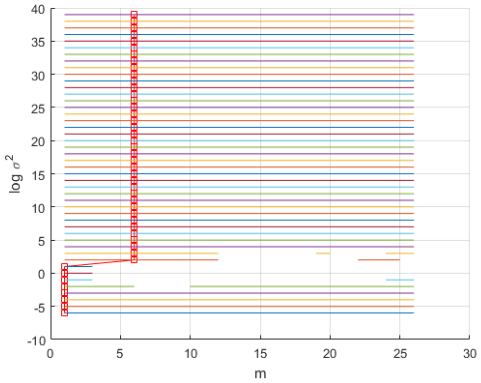
$$\max_{\sigma_k}(|\frac{d(\min(\overline{Z_{s_m}}(m-1 \rightarrow m, \sigma_k)))}{d\sigma_k} + \frac{d(\min(\overline{Z_{s_m}}(m \rightarrow m+1, \sigma_k)))}{d\sigma_k}|) \quad for \quad \sigma_k > \max_{\sigma_k}(\min(\overline{Z_{s_m}}(m, \sigma_k))) \tag{3.8}$$



(a) Plot showing $\overline{Z_{s_m}}$ for different values of σ_k



(b) Side view of plot showing $\overline{Z_{s_m}}$ for different values of σ_k



(c) Top view of plot showing $\overline{Z_{s_m}}$ for different values of σ_k

Figure 3.9: Plots showing $\overline{Z_{s_m}}$ for different values of σ_k from different angles

Figure 3.9a also shows that the overall $\overline{Z_{s_m}}$ corresponding to very small and very large values of σ_k is very small. This is caused by the transformation of the dataset by the kernel function. The values of σ_k with very small overall $\overline{Z_{s_m}}$ result in the transformed data samples being very close to each other (while maintaining the same relative distance) causing the data error Y_{s_m} to be small, resulting in a small estimate of $\overline{Z_{s_m}}$ compared to values of σ_k closer to the peak. This can be seen by comparing Figures 3.3f and 3.3d for dataset S1 by looking at their axes. We are checking for the values of σ_k after the

peak because we know that the data retains its structure even for large values of σ_k as shown in Figures 3.3 and 3.5 but loses its structure (therefore reducing the separability of clusters) quickly as values of σ_k become less than 1. We are trying to find the σ_k resulting in the biggest change in minimum value of $\overline{Z_{s_m}}$ after the peak and we know that for very small values of σ_k the estimated number of clusters $\hat{m}$ is less than CNC as shown in the Figure 3.9a. The minimum $\overline{Z_{s_m}}$ rises as the distance between clusters increases for increasing values of σ_k. For the value of σ_k at which $\hat{m}$ increases to the value of CNC, a rapid decrease in minimum $\overline{Z_{s_m}}$ occurs corresponding to the estimated value of σ_k and the correct clustering result. This is also confirmed by external validation indices which show that the best clustering results are generated for values of σ_k right after the peak.

3.2.6 Time Complexity of Kernel k-MACE

Kernel k-means has computational complexity of $O(N^2 l)$. Calculating the kernelized distance matrix G has a computational complexity of $O(N^2 d)$. The computational complexity of the initialization algorithm (Algorithm 3) is $O(N^2)$. Time complexity of SVD is $O(N^3)$ and the time complexity of PCA is also $O(N^3)$. The time complexity of k-MACE is $O(m') \times O(mNdl)$ [29]. Note that N is the number of samples in the dataset, d is the dimensions of the dataset, l is the number of iterations, m is the number of clusters and $m' = m_{min} - m_{max}$. The total computational complexity of kernel k-MACE is given in Equation 3.9

$$O(\sigma'_k) \times O(N^3) \tag{3.9}$$

Where $\sigma'_k = \sigma_{k_{min}} - \sigma_{k_{max}}$

The pseudo-code for kernel k-MACE algorithm is given below.

Algorithm 4 Kernel k-MACE Algorithm

Require: Estimate the number of clusters $\hat{m}$ and provide a clustering solution.

Input: Data set $\mathbf{x} = [x_1, x_2, ..., x_N]$, range of m $[m_{min}, m_{max}]$ and range of values for $\sigma_k = [\sigma_{k_1}, \sigma_{k_2}, ..., \sigma_{k_{max}}]$ with a fixed interval

Output: Estimated number of clusters $\hat{m}$, the clustering solution $[\hat{C}_1, \hat{C}_2, .., \hat{C}_{\hat{m}}]$ and the optimum value of σ_k

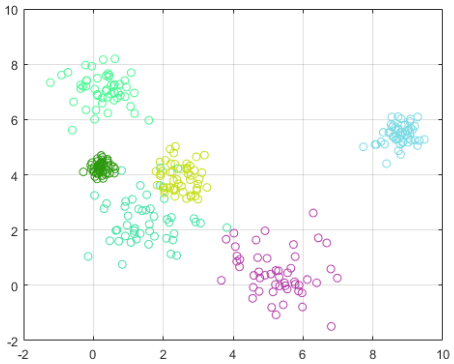
for $(\sigma_k = \sigma_{k_1}; \sigma_k \leq \sigma_{k_{max}}; \sigma_{k_{++}})$ **do**
  for $(m = m_{min}; m \leq m_{max}; m_{++})$ **do**
    $[C_{m1}, C_{m2}, .., C_{mj}] = kernelkmeans(x, m)$ with non-random initial cluster assignments
    for each cluster C_{mj} $j = 1, .., m$ **do**
      Solve cluster compactness y_{smj} of cluster C_{mj} using (3.6)
    end for
    Solve for total cluster compactness Y_{sm} using (1.3)
  end for
  for $(m = m_{min}; m \leq m_{max}; m_{++})$ **do**
    Calculate the covariance of each cluster for each m_i
  end for
  Estimate the optimum $\overline{Z_{s_m}}$ and $\hat{m}$ using the cluster covariances and Y_{s_m} by using equations in Appendix 1
  $[\hat{C}_1, \hat{C}_2, .., \hat{C}_{\hat{m}}] = kernelkmeans(x, \hat{m})$ for each σ_k with non-random initial cluster assignments
end for
Calculate the gradient of the minimum $\overline{Z_{s_m}}$ curve following the condition in (3.8) to obtain the optimum σ_k, the final clustering solution and $\hat{m}$

Chapter 4

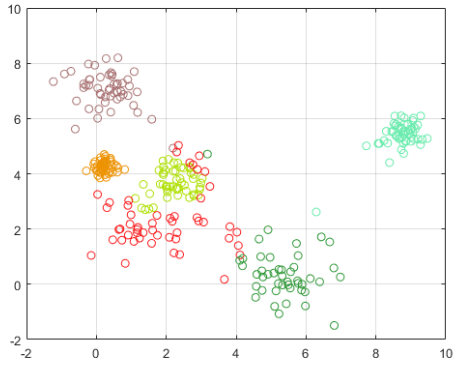
Simulations and Results

4.1 Proposed Cluster Initialization Vs Random Cluster Initialization for Kernel k-means

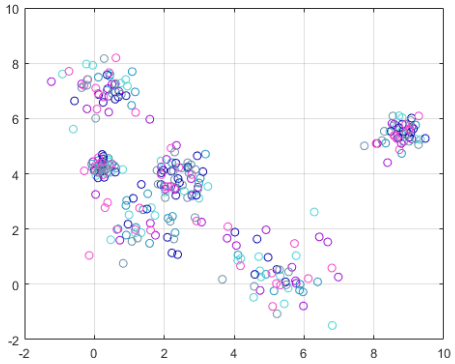
In Section 3.2.2, a new method of initial cluster assignment for data has been proposed. The proposed cluster initialization is shown in Figure 4.1b for dataset E1 in Figure 4.1a. Random cluster initialization is shown in Figure 4.1c. Clustering results for the dataset E1 have been compared for both methods using external validation indices ARI and NVI. Kernel k-means with proposed cluster initialization method results in an ARI and NVI of 0.93 and 0.12 respectively and kernel k-means with random cluster initialization results in ARI and NVI of 0.80 and 0.19 respectively. The results have been obtained over an average of 50 runs.



(a) Synthetic dataset E1



(b) Result of proposed initial cluster assignment algorithm for dataset E1 in Figure 4.1a for m = 6



(c) Result of random cluster assignment for dataset E1 in Figure 4.1a for m = 6

Figure 4.1: Proposed cluster initialization vs random cluster initialization

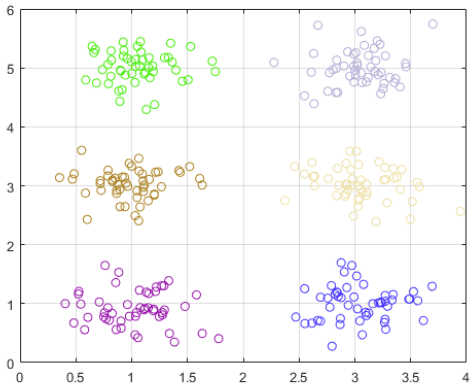
4.2 Synthetic Datasets

Kernel k-MACE is used to cluster synthetic datasets containing Gaussian clusters of different characteristics. We have decided to use Gaussian and Polynomial kernel functions for results, but we use trial and error to obtain the optimum parameter for the Polynomial kernel. Each dataset consists of 300 data samples and has a CNC of 6.

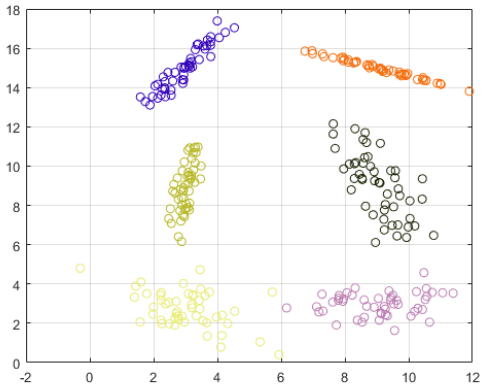
- Dataset S1: Gaussian clusters with uniform variance and fixed cluster center locations with no overlap, shown in Figure 4.2a.
- Dataset S2: Gaussian clusters with varying covariance and fixed cluster center locations with no overlap shown in Figure 4.2b.
- Dataset S3: Gaussian clusters with varying variance and varying cluster center locations with less than 10% overlap shown in Figure 4.2c
- Dataset S4: Gaussian clusters with varying variance, varying cluster center locations and two clusters in close proximity but with no overlap shown in Figure 4.2d
- Dataset S5: Gaussian clusters with varying variance, varying cluster center locations and two clusters with overlap between 10% and 50%, shown in Figure 4.2e
- Dataset S6: Gaussian clusters with varying variance, varying cluster center locations and three overlapping clusters with two clusters with almost 50% overlap shown in Figure 4.2f

Kernel k-MACE was also used to cluster some high dimensional datasets S10 - S15.

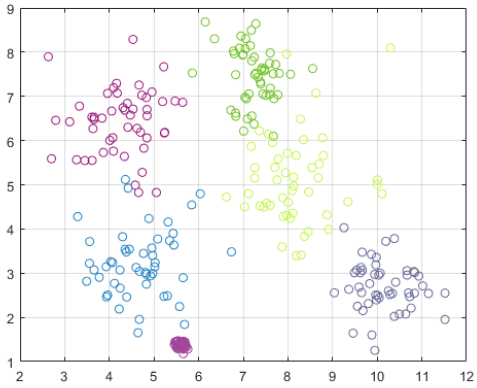
- Dataset S10: Gaussian clusters with uniform variance and 10% chance of overlap
- Dataset S11: Gaussian clusters with varying variance and 10% chance of overlap
- Dataset S12: Gaussian clusters with varying covariance and 10% chance of overlap
- Dataset S13: Gaussian clusters with varying covariance and 50% chance of overlap
- Dataset S14: Gaussian clusters with varying variance and 50% chance of overlap
- Dataset S15: Gaussian clusters with uniform variance and 50% chance of overlap



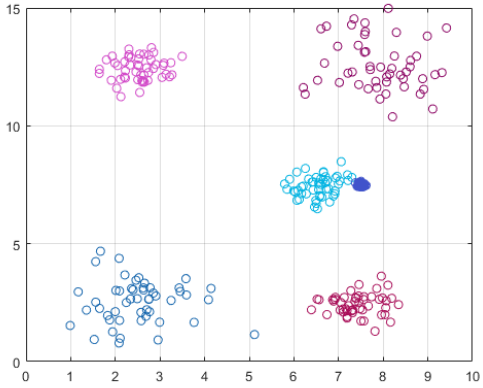
(a) Dataset S1



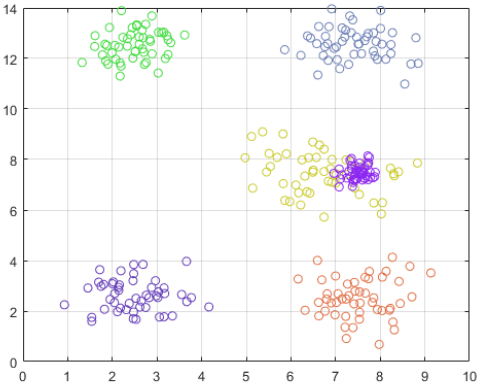
(b) Dataset S2



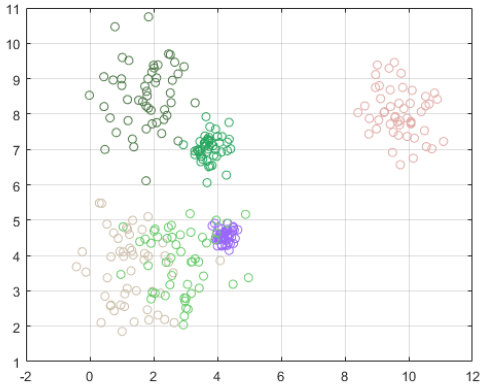
(c) Dataset S3



(d) Dataset S4



(e) Dataset S5



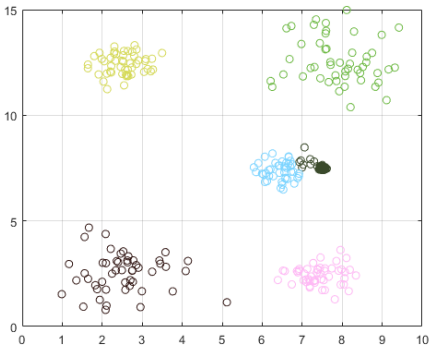
(f) Dataset S6

Figure 4.2: Datasets S1 - S6

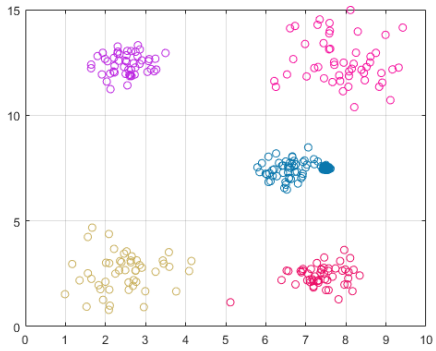
Clustering results from kernel k-MACE have been compared to well known index validity methods such as Gap, Calinski-Harabasz, Davies-Bouldin and Silhouette. These validation indices have been used alongside kernel k-means. Clustering results have also been compared to other well known fully unsupervised clustering methods including G-means and DBSCAN. Kernel based fully unsupervised algorithms proposed in [16], [17], [18] and [19] have not been used for comparison as they do not estimate the optimum value of the kernel parameter and most are computationally very complex. The clustering results for synthetic datasets S1 - S6 are shown in Table 4.1 . $\log\sigma_k^2$ values between -10 and 30 have been used with kernel k-MACE [34]. Each result is generated from an average of 50 runs. ARI and NVI are the external clustering validation indices used to evaluate the results where both range between 0 and 1. An ARI of 1 represents a clustering result matching the true clustering of a given dataset while an NVI of 0 represents clustering result matching the true clustering of a given dataset and vice versa. The best results are in bold font.

Kernel k-MACE [1] is able to identify the correct number of clusters in datasets S1 - S5. Kernel k-MACE with Polynomial kernel is able to identify the correct number of clusters for datasets S1, S2 and S4 - S6. Kernel k-MACE is also able to produce the best clustering result for each dataset. k-MACE is able to correctly identify the correct number of clusters in datasets S1 - S3 but has problem identifying clusters with significantly different variances that overlap or are in very close proximity of each other. As we have shown in Section 3.1, datasets S1 and S2 can be easily clustered in input space. There is some minor overlap between clusters in dataset S3 but k-MACE is able to correctly identify the clusters. The final clustering results for datasets S4, S5 and S6 are shown in Figure 4.3 for both kernel k-MACE as well as k-MACE.

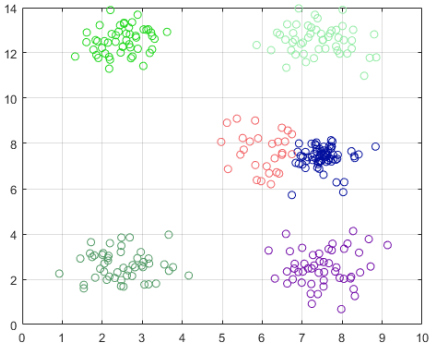
[1]Kernel k-MACE itself only refers to the algorithm that uses the Gaussian kernel function. Use of the algorithm with other kernel functions such as Polynomial will be explicitly stated



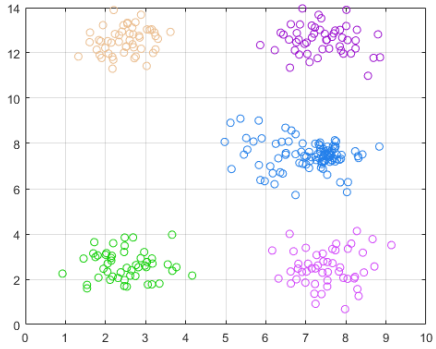
(a) Final clustering result for dataset S4 using kernel k-MACE



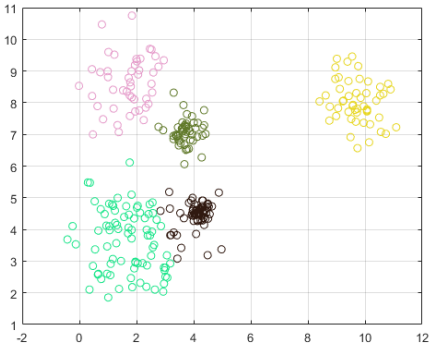
(b) Final clustering result for dataset S4 using k-MACE



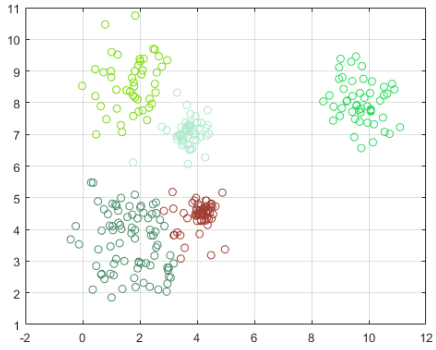
(c) Final clustering result for dataset S5 using kernel k-MACE



(d) Final clustering result for dataset S5 using k-MACE



(e) Final clustering result for dataset S6 using kernel k-MACE



(f) Final clustering result for dataset S6 using k-MACE

Figure 4.3: Final clustering results for datasets S4, S5 and S6 using kernel k-MACE and k-MACE

For both datasets S4 and S5, k-MACE is not able to distinguish between the two overlapping clusters with significantly different variances. k-MACE can identify overlapping clusters that are correlated but it is not able to differentiate between non-correlated clusters that overlap and have significantly different variances [29]. kernel k-MACE on the other hand identifies the 2 distributions as different and is able to correctly identify the clusters. In Section 3.1 we have shown that kernel k-MACE is able to transform Gaussian distributions into kernel space such that they retain the Gaussian profile as in the case of dataset S4. The mere addition of more features to the dataset results in correct clustering being possible for the kernel k-MACE algorithm. Kernel k-MACE is equally efficient at clustering high dimensional gaussian distributions with varying degrees of overlap. This is shown by the clustering results in Table 4.2 (for datasets S10 - S15) as k-MACE produces equally good clustering results as kernel k-MACE for datasets with 15 dimensions and cluster overlap between 10% and 50%.

Kernel k-MACE also estimates the CNC for S5 dataset which has a greater degree of cluster overlap as compared to S4. If the percentage overlap between clusters keeps increasing and the variance of the overlapping clusters becomes similar, eventually kernel k-MACE will fail to distinguish between the clusters in feature space. Dataset S6 is an anomaly and is very hard to find in real life as two clusters have a very high degree of overlap but consist only of two scalar features (two dimensions). Both kernel k-MACE and k-MACE could not identify the correct number of clusters in this dataset.

G-means could not identify the CNC for all of the 2D datasets except S1. DBSCAN was not able to identify the CNC for any 2D dataset. From the internal validation index methods, Davies-Bouldin index worked well for both 2D and high dimensional datasets. However the $std[\hat{m}]$ is large for all 4 methods because of the random initialization of clusters before kernel k-means clustering is performed. Internal validation index methods

produce worse results as compared to kernel k-MACE because they try to optimize a cost function while kernel k-MACE probabilistically calculates the bounds of an unknown error resulting in a better estimation of the number of clusters in a dataset.

Kernel k-MACE with Polynomial kernel is able to identify the CNC for most datasets but does not generate the best clustering results. Polynomial kernel transforms data into feature space such that data samples lie on a hyper-dimensional Polynomial surface governed by the Polynomial kernel parameter. This generates better separation between clusters for some datasets such as S6 as compared to the Gaussian kernel resulting in the estimation of CNC. Kernel k-MACE with polynomial kernel is able to successfully cluster the high dimensional datasets in Table 4.2 just like kernel k-MACE and k-MACE. Polynomial kernel functions work well for clustering datasets containing Gaussian clusters and produce comparable results to the Gaussian kernel function.

Datasets		S1	S2	S3	S4	S5	S6
$\overline{m}$ (CNC)		6	6	6	6	6	6
Kernel k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	5 ± 0
	ARI	**1**	**1**	**0.9**	**0.9**	**0.9**	0.7
	NVI	**0**	**0**	**0.2**	**0.1**	**0.1**	0.3
Kernel k-MACE (Polynomial kernel)	$E[\hat{m}] \pm std[\hat{m}]$	6 ± 0	6 ± 0	5 ± 0	$\mathbf{6 \pm 0}$	6 ± 0	$\mathbf{6 \pm 0}$
	ARI	0.9	0.9	0.7	**0.9**	0.8	**0.7**
	NVI	0.1	0.2	0.3	**0.1**	0.2	**0.4**
k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	6 ± 0	6 ± 0	5 ± 0	5 ± 0	5 ± 0
	ARI	**1**	0.9	0.9	0.8	0.8	0.7
	NVI	**0**	0.1	0.2	0.1	0.1	0.3
G-means	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	12 ± 0	15 ± 0	14 ± 0	12 ± 0	7 ± 0
	ARI	**1**	0.7	0.7	0.7	0.8	0.8
	NVI	**0**	0.2	0.3	0.3	0.2	0.3
DBSCAN	$E[\hat{m}] \pm std[\hat{m}]$	7 ± 0	7 ± 0	7 ± 0	4 ± 0	4 ± 0	9 ± 0
	ARI	0.7	0.7	0.8	0.3	0.3	0.4
	NVI	0.4	0.4	0.2	0.7	0.7	0.5
Kernel k-means + GAP	$E[\hat{m}] \pm std[\hat{m}]$	5.8 ± 1.5	6.4 ± 0.9	6.2 ± 0.8	4.3 ± 1.3	3.9 ± 1.7	5 ± 0.8
	ARI	0.8	0.9	0.8	0.6	0.5	0.6
	NVI	0.2	0.1	0.2	0.3	0.4	0.4
Kernel k-means + Calinski-Harabasz	$E[\hat{m}] \pm std[\hat{m}]$	6.8 ± 1	9.5 ± 3.5	6.8 ± 1.3	5.2 ± 0.8	5.4 ± 1.2	5.9 ± 1.1
	ARI	0.9	0.8	0.8	0.7	0.7	0.7
	NVI	0.1	0.2	0.2	0.2	0.3	0.3
Kernel k-means + Davies-Bouldin	$E[\hat{m}] \pm std[\hat{m}]$	6.1 ± 0.7	7.1 ± 1.2	6 ± 0.6	5.4 ± 1	5 ± 1	5.1 ± 1.6
	ARI	0.8	0.9	0.8	0.7	0.7	0.6
	NVI	0.1	0.1	0.2	0.2	0.2	0.4
Kernel k-means + Silhouette	$E[\hat{m}] \pm std[\hat{m}]$	6.5 ± 0.7	6.8 ± 0.8	6.2 ± 0.6	5.3 ± 1.2	5.7 ± 0.7	5.5 ± 1.6
	ARI	0.9	0.9	0.8	0.7	0.7	0.6
	NVI	0.1	0.1	0.3	0.3	0.2	0.4

Table 4.1: Clustering results for synthetic datasets S1 - S6

Datasets		S10	S11	S12	S13	S14	S15
$\overline{m}$ (CNC)		6	6	6	6	6	6
Kernel k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$
	ARI	**1**	**1**	**1**	**1**	**1**	**1**
	NVI	**0**	**0**	**0**	**0**	**0**	**0**
Kernel k-MACE (Polynomial kernel)	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$
	ARI	**1**	**1**	**1**	**1**	**1**	**1**
	NVI	**0**	**0**	**0**	**0**	**0**	**0**
k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$	$\mathbf{6 \pm 0}$
	ARI	**1**	**1**	**1**	**1**	**1**	**1**
	NVI	**0**	**0**	**0**	**0**	**0**	**0**
G-means	$E[\hat{m}] \pm std[\hat{m}]$	15 ± 0	8 ± 0	9 ± 0	$\mathbf{6 \pm 0}$	13 ± 0	8 ± 0
	ARI	0.7	0.9	0.8	**1**	0.7	0.9
	NVI	0.3	0.1	0.2	**0**	0.3	0.1
DBSCAN	$E[\hat{m}] \pm std[\hat{m}]$	4 ± 0	8 ± 0	7 ± 0	7 ± 0	4 ± 0	12 ± 0
	ARI	0.2	0.1	0.4	0.6	0.3	0.2
	NVI	0.8	0.8	0.5	0.3	0.7	0.8
Kernel k-means + GAP	$E[\hat{m}] \pm std[\hat{m}]$	4.6 ± 1.4	5.4 ± 1.2	8.4 ± 2.4	6.5 ± 0.7	6.1 ± 1.1	4.6 ± 1.5
	ARI	0.7	0.1	0.8	0.2	0.6	0.7
	NVI	0.3	0.9	0.2	0.6	0.5	0.4
Kernel k-means + Calinski-Harabasz	$E[\hat{m}] \pm std[\hat{m}]$	6.7 ± 0.8	6.7 ± 0.7	6.6 ± 1.3	6.8 ± 0.6	6.1 ± 0.4	7.6 ± 2
	ARI	0.9	0.9	0.9	0.9	0.9	0.9
	NVI	0.1	0.1	0.1	0.1	0	0.1
Kernel k-means + Davies-Bouldin	$E[\hat{m}] \pm std[\hat{m}]$	5.9 ± 0.7	5.4 ± 1	5.9 ± 0.6	5.9 ± 0.9	5.5 ± 0.8	5.8 ± 0.8
	ARI	0.9	0.8	0.9	0.9	0.9	0.9
	NVI	0.1	0.2	0.1	0.1	0.1	0.1
Kernel k-means + Silhouette	$E[\hat{m}] \pm std[\hat{m}]$	6.4 ± 0.5	6.4 ± 0.9	6.5 ± 0.6	6.4 ± 0.9	6.9 ± 1.7	6.5 ± 1.1
	ARI	0.9	0.9	0.9	0.9	0.9	0.9
	NVI	0.1	0.1	0.1	0.1	0.1	0.1

Table 4.2: Clustering results for synthetic datasets S10 - S15

4.3 Real Datasets

Real datasets used to evaluate the performance of the kernel k-MACE are given in Table 4.3. The datasets have been obtained from the UCI machine learning repository website [2].

Dataset	Seeds	Iris	Wine	Glass	Soybean	Vertebrate	Thyroid
$\bar{m}$ (CNC)	3	3	3	7	19	3	3
Dimensions	7	4	13	10	35	6	5

Table 4.3: Real Datasets

A brief description of each dataset is given below.

- Seeds: This dataset contains the X-ray imaging data of seeds of 3 different varieties of wheat: Kama, Rosa and Canadian. Internal structure of the seed was visualized using X-ray images.

- Iris: This dataset contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other 2; the latter are not linearly separable from each other.

- Wine: This dataset results from the chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.

- Glass: This dataset contains chemical composition data corresponding to different types of glass containing 7 classes and 10 attributes.

[2] http://archive.ics.uci.edu/ml/

- Soybean: This dataset contains soybean crop data containing 19 classes and 35 attributes.

- Vertebrate: The dataset contains data from patients belonging to one out of three categories: Normal, Disk Hernia or Spondylolisthesis. The dataset contains 3 classes and 6 attributes.

- Thyroid: This dataset contains data from patients belonging to one out of three categories: Normal, Hypothyroidism or Hyperthyroidism. The dataset contains 3 classes and 5 attributes.

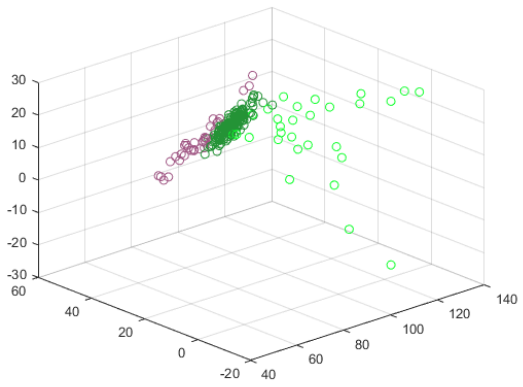
Clustering results for real datasets are shown in Table 4.4. Each result is generated from an average of 50 runs. ARI and NVI are the external clustering validation indices used to evaluate the results. The best clustering results for each dataset are stated in bold font. Kernel k-MACE provides the best estimate of $\hat{m}$ for Seeds, Iris, Wine, Glass and Soybean. A $std[\hat{m}]$ of 0 shows that the method is consistent which is due to the proposed cluster initialization method in Section 3.2.2 which also results in better ARI and NVI values than other methods for most datasets. Kernel k-MACE works well for clustering high dimensional datasets such as Glass and Soybean.

Kernel k-means + GAP index also generates accurate clustering results on average but the $std[\hat{m}]$ is high due to random assignment of data before kernel k-means, resulting in different results each time the algorithm is run. The kernel k-MACE algorithm provides poor clustering result for the Thyroid dataset in terms of ARI and NVI. The first three principal components of the Thyroid dataset have been shown in Figure 4.4a. The two overlapping clusters follow a Gaussian distribution while the third cluster (shown in light green color) cannot be approximated as Gaussian due to the scatter of data samples. Kernel k-MACE assumes that only a Gaussian distribution in the input space

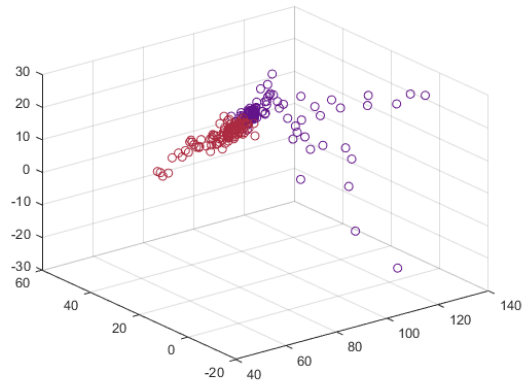
will replicate a Gaussian distribution in feature space, therefore the arbitrary shaped cluster causes one of the two overlapping clusters to merge with it. The final clustering result is shown in Figure 4.4b. G-means is not able to estimate the correct number of clusters for any real dataset in Table 4.4. DBSCAN results in the correct number of clusters for Iris dataset but totally fails to obtain any clustering result for Thyroid and Vertebral datasets.

The high accuracy of $E[\hat{m}]$ of kernel k-means + GAP and kernel k-means + Davies-Bouldin for the soybean and glass datasets strengthens the argument that kernel k-means is able to find more accurate clustering partitions for real datasets as compared to k-means based methods including k-MACE [29].

Kernel k-MACE with Polynomial kernel produces inferior results as compared to kernel k-MACE for most real datasets. However kernel k-MACE with polynomial kernel is able to estimate the CNC for the Vertebral dataset. This shows that the Gaussian kernel function is better for clustering real datasets as compared to the Polynomial kernel function.



(a) Plot of first three principal components of the Thyroid dataset



(b) Plot of first three principal components of the kernel k-MACE clustering solution for Thyroid dataset

Datasets		Seeds	Iris	Wine	Glass	Soybean	Vertebral	Thyroid
$\overline{m}(CNC)$		3	3	3	7	19	3	3
Kernel k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{3 \pm 0}$	$\mathbf{3 \pm 0}$	$\mathbf{3 \pm 0}$	$\mathbf{7 \pm 0}$	$\mathbf{18 \pm 0}$	2 ± 0	2 ± 0
	ARI	**0.7**	**0.8**	**0.4**	**0.2**	**0.4**	0.3	0.1
	NVI	**0.5**	**0.3**	**0.7**	**0.7**	**0.4**	0.7	0.9
Kernel k-MACE (Polynomial kernel)	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{3 \pm 0}$	3 ± 0	2 ± 0	4 ± 0	16 ± 0	$\mathbf{3 \pm 0}$	2 ± 0
	ARI	**0.7**	0.7	0.3	0.2	0.4	**0.1**	0.1
	NVI	**0.5**	0.5	0.7	0.7	0.5	**0.9**	0.9
k-MACE	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{3 \pm 0}$	3 ± 0	$\mathbf{3 \pm 0}$	5 ± 0	12 ± 0	2 ± 0	2 ± 0
	ARI	**0.7**	0.7	**0.4**	0.2	0.5	0.3	0.1
	NVI	**0.5**	0.4	**0.7**	0.8	0.4	0.7	0.9
G-means	$E[\hat{m}] \pm std[\hat{m}]$	4 ± 0	4 ± 0	2 ± 0	11 ± 0	16 ± 0	5 ± 0	7 ± 0
	ARI	0.3	0.5	0.2	0.1	0.3	0.3	0.1
	NVI	0.7	0.6	0.7	0.9	0.8	0.8	0.9
DBSCAN	$E[\hat{m}] \pm std[\hat{m}]$	2 ± 0	3 ± 0	1 ± 0	2 ± 0	4 ± 0	1 ± 0	1 ± 0
	ARI	0	0.5	0	0.2	0.1	0	0
	NVI	1	0.6	1	0.8	0.5	1	1
Kernel k-means + GAP	$E[\hat{m}] \pm std[\hat{m}]$	2.9 ± 0.6	3.2 ± 0.6	2.5 ± 4.5	6.4 ± 1.5	17.8 ± 3.7	1.7 ± 2.2	8.5 ± 3.6
	ARI	0.6	0.7	0	0.2	0.4	0	0.2
	NVI	0.5	0.4	1	0.8	0.5	1	0.8
Kernel k-means + Calinski-Harabasz	$E[\hat{m}] \pm std[\hat{m}]$	$\mathbf{3 \pm 0}$	3.4 ± 0.5	9.6 ± 7	2.4 ± 0.5	4.1 ± 2.3	2.2 ± 0.4	4.7 ± 2.9
	ARI	**0.7**	0.7	0	0.2	0.2	0.2	0.1
	NVI	**0.5**	0.4	1	0.8	0.7	0.9	0.9
Kernel k-means + Davies-Bouldin	$E[\hat{m}] \pm std[\hat{m}]$	2 ± 0	2 ± 0	3.3 ± 2.7	6.9 ± 3	15.9 ± 4.1	2.6 ± 2	5.4 ± 1.8
	ARI	0.5	0.6	0	0.2	0.4	0.2	0.3
	NVI	0.6	0.4	1	0.8	0.5	0.9	0.8
Kernel k-means + Silhouette	$E[\hat{m}] \pm std[\hat{m}]$	2.1 ± 0.3	2 ± 0	2 ± 0	2.5 ± 0.8	14 ± 3	2.3 ± 0.7	$\mathbf{3.8 \pm 0.9}$
	ARI	0.5	0.6	0	0.2	0.4	0.2	**0.4**
	NVI	0.6	0.4	1	0.8	0.5	0.9	**0.8**

Table 4.4: Clustering results for real datasets

4.4 Normality Tests for Kernel k-MACE Evaluation

Normality tests can be used to show that the clustering result generated by kernel k-MACE will consist of clusters following a Gaussian distribution whether the dataset being clustered consists of clusters following a Gaussian distribution or an unknown distribution (arbitrary shaped clusters). The final clusters resulting from kernel k-MACE clustering have been tested to check if they are normally distributed. The following normality tests have been used [35][36][37].

- Jarque-Bera test
- Anderson-Darling test
- Lilliefors test

Each test evaluates a single dimension (feature) of a cluster at a time to check if the hypothesis (data follows a normal distribution with an unknown mean and variance) is true or false. For real datasets in Table 4.3 which are not generated from Gaussian distributions, results from normality tests show that the clusters resulting from kernel k-MACE follow a Gaussian distribution with at least 90% probability.

Chapter 5

Conclusion and Future Work

This thesis presents a kernel based clustering scheme which estimates the number of clusters in a dataset while simultaneously estimating the value of the Gaussian kernel parameter corresponding to the correct clustering result. Kernel functions transform data into feature space which is governed by the value of the kernel function parameter. Basics of kernel functions, kernel methods and some internal validation indices which can be used as fully unsupervised clustering methods were discussed in Chapter 2.

Kernel k-MACE clustering scheme is presented in Chapter 3. Kernel k-MACE probabilistically estimates the number of clusters for a given dataset in while also obtaining the value of the Gaussian kernel parameter σ_k corresponding to the best clustering result. Kernel k-MACE can be divided into three sections: Initialization and clustering of data, cluster evaluation and estimation of the optimum value of σ_k. A cluster initialization technique has been proposed to improve the results of kernel k-means which is the algorithm used for clustering data in feature space. k-MACE is then used to evaluate the clustering result in feature space. Finally the optimum value of the Gaussian kernel parameter is estimated from a given range, which is a major contribution of this research.

The feature map of data is visualized and the effects of different values of σ_k on the feature map are shown. Kernel k-MACE has a higher time complexity as compared to k-MACE which is caused by the decomposition of the kernelized distance matrix using SVD.

Simulations and results are provided in Chapter 4. The proposed cluster initialization method improves the clustering results of kernel k-means when compared to random cluster initialization. Results from synthetic datasets show that kernel k-MACE is able to successfully cluster datasets containing overlapping Gaussian clusters with significantly different variances and outperforms the methods used for comparison. This is the result of transformation of data into higher dimensions resulting in a better separation of clusters. However as the percentage of overlap increases beyond 40% and the variances become similar, kernel k-MACE is not able to identify the clusters correctly. Kernel k-MACE with polynomial kernel on the other hand is able to identify overlapping clusters with 50% overlap outperforming kernel k-MACE for this specific case. Kernel k-MACE outperforms methods used for comparison for most real datasets. Kernel k-MACE produces consistent clustering results due to the use of proposed cluster initialization with kernel k-means. Normality tests are used to evaluate the clustering results from kernel k-MACE to show that the clusters are normally distributed and are independent of the distribution of the original data.

The proposed method calculates the value of σ_k parameter for the Gaussian kernel function using a non-generalizable method. An avenue for future research would be to analytically estimate the best value of σ_k. Applications of kernel methods in clustering show that they are most commonly used for clustering data which cannot be separated in input space i.e. the data is non-linearly separable or arbitrary shaped. An interesting topic for future research would be to extend kernel k-MACE for correctly estimating the

number of clusters in an arbitrary shaped dataset containing non-linearly separable data. Currently this cannot be done as the probabilistic estimation of the number of clusters uses the assumption that the clusters in the dataset follow a Gaussian distribution. Kernel k-MACE has been used with Gaussian and Polynomial kernel functions as Sigmoid kernel requires the optimization of two parameters and therefore has not been used. Another avenue for future research would be to extend kernel k-MACE to work with the Sigmoid kernel and find its optimum parameters which should be straightforward.

Appendix 1

k-MACE

Average Central Error in k-MACE is given by Equations (1.1) and (1.2)

$$Z_{s_m} = \frac{1}{N} \sum_{j=1}^{m} Z_{s_{mj}} \tag{1.1}$$

and

$$Z_{s_{mj}} = \left\| \overline{c}_{x_{mj}} - c_{mj} \right\|_F^2 \tag{1.2}$$

Data Error in k-MACE is given by (1.3) and (1.4)

$$Y_{s_m} = \frac{1}{N} \sum_{j=1}^{m} Y_{s_{mj}} \tag{1.3}$$

and

$$Y_{s_{mj}} = \| x_{mj} - c_{mj} \|_F^2 \tag{1.4}$$

$\overline{Z_{s_m}}$ is estimated by using Y_{s_m} for each clustering result corresponding to the provided range of values of m. The expected value and variance of Y_{s_m} are given by Equations

(1.5) and (1.6) respectively.

$$E[Y_{s_m}] = \frac{1}{N}\sum_{j=1}^{m} E[Y_{s_{mj}}] \tag{1.5}$$

$$Var[Y_{s_m}] = \frac{1}{N^2}\sum_{j=1}^{m} Var[Y_{s_{mj}}] \tag{1.6}$$

The probabilistic bounds of $\Delta_{S_{mj}}$ in Equation (1.7) are given by Equation (1.8). These are used to calculate the bounds of $E[Z_{s_{mj}}]$ and $Var[Z_{s_{mj}}]$ in Equations (1.12), (1.13) and (1.14).

$$\underline{\|\Delta_{S_{mj}}\|_F^2} \leq \|\Delta_{S_{mj}}\|_F^2 \leq \overline{\|\Delta_{S_{mj}}\|_F^2} \tag{1.7}$$

$$\begin{aligned} \overline{\|\Delta_{S_{mj}}\|_F^2} &= x_{S_{mj}} + k_{S_{mj}} \\ \underline{\|\Delta_{S_{mj}}\|_F^2} &= x_{S_{mj}} - k_{S_{mj}} \end{aligned} \tag{1.8}$$

$$\begin{aligned} x_{S_{mj}} &= (m_{mj} - k_{S_{mj}}) - \alpha_N^2 \frac{2}{dn_j} \sum_{x_i \in C_{mj}} tr(\overline{\wedge_{x_i}}) \\ where \qquad m_{wj} &= \frac{n_{mj}-1}{n_{mj}} \sum_{x_i \in C_{mj}} tr(\overline{\wedge_{x_i}}) \end{aligned} \tag{1.9}$$

$$k_{S_{mj}} = 2\alpha_N \left[v_{S_{ml}} + \left(\frac{(4\alpha_N^2 + 2d^2)}{d^2 n_{mj}^2} \sum_{x_i, x_k \in C_{mj}, i \neq k} tr(\overline{\wedge}_{x_i}, \overline{\wedge}_{x_k}) \right) + \left(\frac{(4\alpha_N^2 + 2d^2 n_{mj}^2)}{d^2 n_{mj}^2} \sum_{x_i \in C_{mj}} tr((\overline{\wedge}_{x_i})^2) \right) \right]^{1/2} \tag{1.10}$$

$$v_{S_{mj}} = \frac{-4(m_{wj} - y_{S_{mj}})}{dn_{mj}} \sum_{x_i \in C_{mj}} tr(\overline{\wedge}_{x_i}) \tag{1.11}$$

$$\overline{E[Z_{s_{mj}}]} \leq \overline{\|\Delta_{S_{mj}}\|_F^2} + \frac{1}{n_{mj}} \sum_{i=1}^{n_{mj}} tr(\overline{\wedge}_{x_{mj}^i}) \tag{1.12}$$

$$\underline{E[Z_{s_{mj}}]} \leq \underline{\|\Delta_{S_{mj}\|_F^2}} + \frac{1}{n_{mj}} \sum_{i=1}^{n_{mj}} tr(\overline{\wedge}_{x_{mj}^i}) \tag{1.13}$$

$$\begin{aligned} Var[Z_{s_{mj}}] = \frac{2}{n_{mj}^2} \sum_{x_i \in C_{mj}} tr((\overline{\wedge}_{x_i})^2) + \\ \frac{2}{n_{mj}^2} \sum_{x_i, x_k \in C_{mj}, i \neq k} tr(\overline{\wedge}_{x_i}, \overline{\wedge}_{x_k}) \end{aligned} \tag{1.14}$$

The expected value and variance of Z_{s_m} can be estimated using Equations (1.15) and (1.16).

$$E[Z_{s_m}] = \frac{1}{N} \sum_{j=1}^{m} E[Z_{s_{mj}}] \tag{1.15}$$

$$Var[Z_{s_m}] = \frac{1}{N^2} \sum_{j=1}^{m} Var[Z_{s_{mj}}] \tag{1.16}$$

The probabilistic bounds of Z_{s_m} can then be calculated using Equations (1.17) and (1.18)

$$\overline{Z_{s_m}} = E[Z_{s_m}] + \beta_N \sqrt{var[Z_{s_m}]} \tag{1.17}$$

$$\underline{Z_{s_m}} = E[Z_{s_m}] - \beta_N \sqrt{var[Z_{s_m}]} \tag{1.18}$$

Bibliography

[1] H. Zhang, B. Pang, K. Xie, H. Wu. "An Efficient Algorithm for Clustering Search Engine Results", *International Conference on Computational Intelligence and Security*, 2006.

[2] Sara Dolnicar. "Data-driven Market Segmentation in Tourism - Approaches, Changes Over Two Decades and Development Potential", *15th International Research Conference of the Council for Australian University Tourism and Hospitality Education*, pp. 346-360, 2006.

[3] Katherine Samuelowicz, John D. Bain. "Revisiting Academics Beliefs about Teaching and Learning", *Higher Education*, pp. 299-325, 2001.

[4] H. G. Wilson, B. Boots, A. A. Millward. "A Comparison of Hierarchical and Partitional Clustering Techniques for Multispectral Image Classification", *IEEE International Geoscience and Remote Sensing Symposium*, 2002.

[5] Ozer Sedat, Chen Chi H., Cirpan Hakan A. "A Set of new Chebyshev Kernel Functions for Support Vector Machine Pattern Classification", *Pattern Recognition*, vol 44, pp. 1435-1447, 2011.

[6] Liu Yi-Hung, Wu Chien-Te, Cheng Wei-Teng, Hsiao Yu-Tsung, Chen Po-Ming, Teng Jyh-Tong. "Emotion Recognition from Single-trial EEG based on Kernel Fisher's

Emotion Pattern and Imbalanced Quasiconformal Kernel Support Vector Machine", *Sensors (Basel, Switzerland)*, vol 14, pp. 13361-13388, 2014.

[7] Radha Chitta. "Kernel-Based Clustering of Big Data", *PhD thesis. Michigan State University*, 2015.

[8] Schlkopf, Bernhard, Smola, Alexander, Mller, Klaus-Robert. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem", *Neural Computation*, vol 10, pp. 1299-1319, 1998.

[9] Sarma T. H., Viswanath P., Reddy B. E. "A Fast Approximate Kernel k-means Clustering Method for Large Data Sets", *IEEE Recent Advances in Intelligent Computational Systems*, pp. 545-550, 2011.

[10] Tzortzis G. F., Likas A. C. "The Global Kernel k-Means Algorithm for Clustering in Feature Space", *IEEE Transactions on Neural Networks*, vol 20, pp. 1181-1194, 2009.

[11] T Hitendra Sharma, P Viswanath, B Eswara Reddy. "Single Pass Kernel k-means Clustering Method", *Indian Academy of Sciences*, vol. 38, part 3, pp. 407-419, 2013.

[12] Nikolaos Tsapanos, Anastasios Tefas, Nikolaos Nikolaidis, Ioannis Pitas. "A Distributed Framework for Trimmed Kernel k-Means Clustering", *Pattern Recognition*, vol. 48, pp. 2685-2698, 2015.

[13] Lujiang Zhang, Xiaohui Hu. "Locally Adaptive Multiple Kernel Clustering", *Neurocomputing*, vol. 137, pp. 192-197, 2014.

[14] Chiheb-Eddine Ben NCir, Nadia Essoussi, Mohamed Limam. "Kernel-Based Methods to Identify Overlapping Clusters with Linear and Nonlinear Boundaries", *Journal of Classification*, vol. 32, pp. 176-211, 2015.

[15] A. Lorette, X. Descombes, J. Zerubia. "Fully Unsupervised Fuzzy Clustering with Entropy Criterion", *Proceedings 15th International Conference on Pattern Recognition*, vol. 3, pp. 986-989, 2000.

[16] R.J. Kuo, Y.D. Huang, Chih-Chieh Lin, Yung-Hung Wud, Ferani E. Zulvia. "Automatic Kernel Clustering with Bee Colony Optimization Algorithm", *Information Sciences*, vol. 283, pp. 107-122, 2014.

[17] Lei Zhang, Qixin Cao. "A Novel Ant-based Clustering Algorithm using the Kernel Method", *Information Sciences*, vol. 181, pp. 4658-4672, 2011.

[18] Hong Jia, Yiu-ming Cheung, Jiming Liu. "Cooperative and Penalized Competitive Learning with Application to Kernel-based Clustering", *Pattern Recognition*, vol. 47, pp. 3060-3069, 2014.

[19] Mark Girolami. "Mercer Kernel-Based Clustering in Feature Space", *IEEE transactions on neural networks*, vol. 13, no. 3, 2002.

[20] Tibshirani, Robert, Walther, Guenther, Hastie, Trevor. "Estimating the Number of Clusters in a Data Set via the Gap Statistic", *Journal of the Royal Statistical Society*, vol. 63, pp. 411-423, 2001.

[21] T. Calinski, J. Harabasz. "A Dendrite Method for Cluster Analysis", *Communications in Statistics*, vol. 3, no.1, pp. 1-27, 1974.

[22] D. L. Davies, D. W. Bouldin. "A Cluster Separation Measure", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224-227, 1979.

[23] L. Kaufman, P.J. Rousseeuw. "Finding Groups in Data: An Introduction to Cluster Analysis", New York, NY: Wiley, 1990.

[24] Camps-Valls Gustavo, Rojo-lvarez Jos L., Martnez-Ramn Manel. "Kernel Clustering for Knowledge Discovery in Clinical Microarray Data Analysis in Chapter 3: Kernel Methods in Bioengineering, Signal and Image Processing", *Idea Group Pub*, 2007

[25] Dunn, J. C. "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters", *Journal of Cybernetics*, pp. 32-57, 1973

[26] Kaufman, L. and Rousseeuw, P.J. "Clustering by means of Medoids, in Statistical Data Analysis Based on the L_1Norm and Related Methods", *North-Holland*, pp. 405-416, 1987

[27] Lujiang Zhang, Xiaohui Hu. "Locally Adaptive Multiple Kernel Clustering", *Neurocomputing, Elsevier B.V.*, vol. 137, pp. 192-197, 2014

[28] Alona Golts, Michael Elad. "Linearized Kernel Dictionary Learning", *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, 2016

[29] Edward Wyndel Nidoy. "k-MACE Clustering for Gaussian Clusters", *MASc thesis. Ryerson University*, 2016

[30] Michael Eigensatz. "Insights into the Geometry of the Gaussian Kernel and an Application in Geometric Modeling", *Master's thesis. Swiss Federal Institute of Technology Zurich*, 2006

[31] Lech Szymanski, Brendan McCane. "Visualising Kernel Spaces", *Image and Vision Computing New Zealand (IVCNZ)*, pp. 449-452, 2011

[32] K. A. Abdul Nazeer, M. P. Sebastian. "Improving the Accuracy and Efficiency of the k-means Clustering Algorithm", *Proceedings of the World Congress on Engineering*, vol 1, 2009

[33] Edward Nidoy, Soosan Beheshti. "k-MACE Clustering", *IEEE Transactions on Signal Processing*, Submitted for Publication, 2017

[34] Kuo-Ping Wu, Sheng-De Wang. "Choosing the Kernel Parameters for Support Vector Machines by the Inter-cluster Distance in the Feature Space", *Pattern Recognition, Elsevier*, vol 42, pp. 710-717, 2009

[35] Carlos M. Jarque, Anil K. Bera. "Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals", *Economics Letters*, pp. 255-259, 1980

[36] T. W. Anderson, D. A. Darling. "Asymptotic Theory of certain Goodness of Fit Criteria based on Stochastic Processes", *Ann. Math. Stat*, pp. 193-212, 1952

[37] Lilliefors, H. "On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown", *Journal of the American Statistical Association*, vol 62, pp. 399-402, 1967