3-10-2021

Efficient Clustering via Kernel Principal Component Analysis and Optimal One-dimensional Thresholding

Nachiket Bhide
University of Windsor

Efficient Clustering via Kernel Principal Component Analysis and Optimal One-dimensional Thresholding

By

Nachiket Bhide

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2021

Efficient Clustering via Kernel Principal Component Analysis and Optimal One-dimensional Thresholding

by

Nachiket Bhide

APPROVED BY:

D. Yang
Department of Mathematics and Statistics

A. Ngom
School of Computer Science

L. Rueda, Advisor
School of Computer Science

January 21, 2021

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

I. Co-Authorship

I hereby declare that this thesis incorporates material that is result of joint research, as follows:

Chapter 2 of the thesis was co-authored with Luis Rueda. L. Rueda contributed with initial thoughts about this research area and the main ideas, and assisted in elaborating on the new novel approach implemented in this work. Both authors contributed in finalizing the idea and follow-up discussions. N. Bhide implemented the method, data pre-processing, experimental design, and the data analysis. N. Bhide wrote the contents of the chapter. L. Rueda assisted in writing and reviewing the content of the chapter.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis. I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publication

This thesis includes an original paper that has been previously submitted for publication in the following journal:

Thesis chapter	**Publication title**	**Publication Status**
Chapter 2	Nachiket Bhide and Luis Rueda, Efficient Clustering via Kernel Principal Component Analysis and Optimal One-dimensional Thresholding, 2020	Submitted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution. I understand that my thesis may be made electronically available to the public.

ABSTRACT

Several techniques are used for clustering of high-dimensional data. Traditionally, clustering approaches are based on performing dimensionality reduction of high-dimensional data followed by classical clustering such as k-means in lower dimensions. However, this approach based on k-means does not guarantee optimality. Moreover, the result of k-means is highly dependent on initialization of cluster centers and hence not repeatable, while not being optimal. To overcome this drawback, an optimal clustering approach in one dimension based on dimensionality reduction is proposed. The one-dimensional representation of high dimensional data is obtained using Kernel Principal Component Analysis. The one-dimensional representation of the data is then clustered optimally using a dynamic programming algorithm in polynomial time. Clusters in the one-dimensional data are obtained by minimizing the sum of within-class variance while maximizing the sum of between-class variance. The advantage of the proposed approach is demonstrated on synthetic and real-life datasets over standard k-means in terms of optimality and repeatability.

DEDICATION

Dedicated to my parents for their endless love, support and encouragement.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Luis Rueda, for his exceptional guidance, patience, and constant support. This thesis would not have been possible without his ideas and suggestions. I have greatly benefited from his knowledge and it has been a good learning experience for me to work under his supervision. I also took his Pattern Recognition course and his teaching was very helpful in understanding machine learning concepts in detail.

I would also like to thank my external reader Dr. Dilian Yang and and internal reader Dr. Alioune Ngom for being a part of this thesis. Their feedback and comments have helped me in further improvement of this thesis. I also had taken Dr.Ngom's data science course during my Masters and it was a great learning experience. His teaching helped me to understand basic concepts in data science and statistical methods.

I would like to thank the School of Computer Science, the graduate program secretary Christine Weisener and all other department staff for their help and support. Finally, I would like to thank my friends Musab Naik and Saiteja Danda for their continuous encouragement and support throughout my Masters.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

PCA	Principal Component Analysis
NLDR	Non-Linear Dimensionality Reduction
KPCA	Kernel Principal Component Analysis
RBF	Radial Basis Function

CHAPTER 1

Introduction

1.1 Unsupervised Learning and Clustering

Unsupervised learning is one of the main categories in machine learning along with supervised learning and reinforcement learning. In supervised learning, machine learning models are developed to perform classification or regression tasks using a training dataset. In order to learn and tune the parameters of a machine learning model, the training dataset is used which consists of a set of known class labels or outcomes corresponding to the input data. Unlike supervised machine learning approaches, unsupervised learning techniques infer previously undetected patterns from data without considering any training dataset. Unsupervised learning approaches can be used to discover the underlying structure of data. Also, the cost associated with collecting, labeling and creating training datasets is avoided. The applications of unsupervised learning techniques include clustering, anomaly detection, association mining and latent variable models, among others.

Clustering is the task of grouping the data in such a way that objects in one cluster are more similar to each other than those in other clusters. Similarity among data points can be measured using one or more parameters such as distances among data points, density of data points or based on other statistical distributions. Clustering therefore can be defined as a multi-objective optimization problem. There are different types of clustering algorithms such as connectivity based such as hierarchical clustering, centroid-based such as k-means algorithm, density based such as DBSCAN, graph based and others.

1.2 k-means Clustering

k-means is one of the most widely-used clustering algorithms. It is a centroid based clustering approach in which a cluster is identified by its centroids. It partitions data into k clusters such that each data point belongs to the cluster with the nearest mean or cluster centroid. Let $(\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, ..., \mathbf{x_n})$ be a set of n data points where each data point is d-dimensional vector. Let $C = \{C_1, C_2, C_3, ...C_k\}$ represent k clusters ($k \leq n$) into which the data is to be partitioned. The k-means algorithm tries to cluster the data such that the within-cluster variance is minimized. The objective function of k-means can be expressed as follows:

$$\min \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \boldsymbol{\mu_i})^2, \tag{1}$$

where vector $\boldsymbol{\mu_i}$ is the mean of the data points in cluster C_i.

The most common form of the k-means algorithm uses an iterative approach and is also referred to as Lloyd's algorithm[6]. It works by alternating between two steps, namely the assignment step and the update step. The k-means algorithm beings by initializing k random centroids which correspond to k clusters in which the data is to be partitioned. In the first step of the assignment, the data points are assigned to the cluster of the nearest centroid. In the second step, the cluster centroids are updated based on the assignment of the data points in the previous step. The assignment step and the update step are repeated with the newly updated cluster centroids. This iterative process continues until convergence; that is, when there are no new changes in the assignment of data points. However, k-means may fall into a local optimum and does not guarantee optimality. The distance function used in k-means is the squared Euclidean distance and hence it is not suitable for clustering high dimensional data wherein Euclidean distances do not necessarily represent the similarity between data points. Examples of these are vertices or edges in a graph, bags of words, sets of documents, and many others.

1.3 Dimensionality Reduction

Dimensionality reduction refers to the process of transforming high-dimensional data into a low-dimensional representation such that the new representation retains meaningful properties and characteristics of the original high-dimensional dataset. In high-dimensional spaces, machine learning algorithms often suffer from problems such as curse of dimensionality. Also, in many cases, high-dimensional features of the data cannot be represented using Euclidean distance and hence algorithms such k-means clustering which are based on Euclidean distances do not perform well in high dimensional spaces. There are two main types of dimensionality reduction techniques, namely, linear dimensionality reduction techniques such as Principal Component Analysis (PCA) and Nonlinear dimensionality reduction techniques such as Kernel Principal Component Analysis (KPCA), Isomaps, Locally Linear Embedding (LLE), Self-organizing maps (SOM) and so on. Linear dimensionality reduction techniques are used to transform high dimensional data which is linearly separable into lower dimensions. However, when high dimensional data is linearly inseparable, non-linear dimensionality reduction techniques are used. Data is considered to lie on an embedded non-linear manifold in higher dimensions which is then projected onto lower dimensions using non-linear dimensionality techniques such as manifold learning.

1.3.1 Principal Component Analysis

PCA is a widely-used technique that performs dimensionality reduction by obtaining orthogonal projections of high-dimensional data onto a lower-dimensional space, such that the dispersion of the projected data in terms of the eigenvectors of the within-class scatter is maximized. Consider a high-dimensional data set $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x_i}$ represents a single data point in the d-dimensional Euclidean space. PCA is a linear projection method from the d-dimensional input space to the p-dimensional output space ($p <<< d$) by solving the eignevalue and eigenvector problem as follows:

$$\mathbf{Cv} = \lambda \mathbf{v}, \tag{2}$$

where $\mathbf{C}$ is the covariance matrix of the centered data:

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i}).(\mathbf{x_i})^T, \tag{3}$$

with λ and $\mathbf{v}$ being the eigenvalues and eigenvectors of $\mathbf{C}$, respectively. Let $W = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p\}$ be the matrix of the p largest eigenvalues corresponding to the largest eigenvectors arranged into columns. The principal components $\mathbf{Y}$ are defined as:

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X} \tag{4}$$

The new principal axes capture the maximum variance, such that the data projected on the new axes are uncorrelated. However, classical PCA does not take into account the non-linear relationships among high-dimensional input data points [5]. To overcome this problem, kPCA, introduced in [9], is widely used to extract non-linear, high-dimensional features. Figure 1.3.1 illustrates the concept of PCA [1].

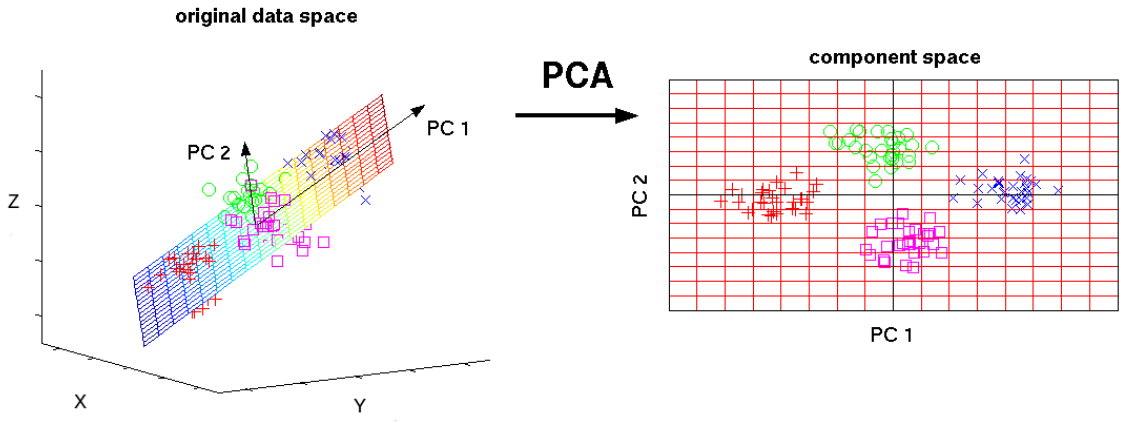

Fig. 1.3.1: Example of principal component analysis.

1.3.2 Kernel Principal Component Analysis

Non-linearity is introduced by mapping the data from the input space R^N to a feature space F. As per Cover's theorem, nonlinear data in the input space is more likely to be linear after high-dimensional nonlinear mapping [4]. In kPCA, a nonlinear kernel

function is used instead of the standard dot product, which implicitly performs PCA in the high-dimensional space F. Therefore, kPCA is able to produce features that capture nonlinear structures in the data more efficiently than linear PCA.

The mapping function is defined as follows:

$$\begin{aligned} \phi : R^N &\longrightarrow F \\ \mathbf{x_i} &\longrightarrow \phi(\mathbf{x_i}) \end{aligned} \tag{5}$$

and the correlation matrix in the feature space F is defined as follows:

$$\tilde{\mathbf{C}} = \frac{1}{N} \sum_{i=1}^{N} \phi(\mathbf{x_i}) \phi(\mathbf{x_i})^T, \tag{6}$$

kPCA is based on solving the eigenvector problem in the transformed space:

$$\tilde{\mathbf{C}} \tilde{\mathbf{v}} = \tilde{\lambda} \tilde{\mathbf{v}}, \tag{7}$$

where $\tilde{\lambda}$ and $\tilde{\mathbf{v}}$ are the corresponding eigenvalues and eigenvectors of $\tilde{\mathbf{C}}$, respectively. $\tilde{\mathbf{v}}$ lies in the span of $\phi(\mathbf{x_1}), \ldots, \phi(\mathbf{x_N})$ and is therefore a linear combination of $\phi(\mathbf{x_i})$ elements. It can be written as follows:

$$\tilde{\mathbf{v}} = \sum_{j=1}^{N} a_j \phi(\mathbf{x_j}) \tag{8}$$

The kernel trick allows to find the corresponding eigenvalues and eigenvectors without explicitly mapping the data onto space F, which may be of infinite dimension, for example, in the RBF kernel. For this purpose, the kernel is defined as follows:

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\phi(\mathbf{x_i})^T \phi(\mathbf{x_j})) \tag{9}$$

To extract the principal components of any point $\mathbf{x}$, the image $\phi(\mathbf{x})$ of the point needs to be projected onto the P obtained eigenvectors. Eigenvectors $\{\tilde{\mathbf{y}}_\mathbf{1}, \ldots, \tilde{\mathbf{y}}_\mathbf{p}, \ldots, \tilde{\mathbf{y}}_\mathbf{P}\}$ are the non-linear principal components in the feature space F. This can be expressed

mathematically as follows:

$$\tilde{\mathbf{y}}_{\mathbf{p}} = \tilde{\mathbf{v}}_{\mathbf{p}}^{\mathbf{T}} \phi(\mathbf{x}) = \sum_{i=1}^{N} a_{pi} K(\mathbf{x_i}, \mathbf{x}) \tag{10}$$

Fig. 1.3.2 shows the original dataset which is not linearly separable, while Fig.1.3.3 shows the result of KPCA using the radial basis function (RBF) kernel with a specific gamma parameter. Applying k-means to the data of Figure 1.3.2 will never capture the two groups of points accurately, while choosing the right initialization parameters, k-means will be able to distinguish between the two groups accurately.

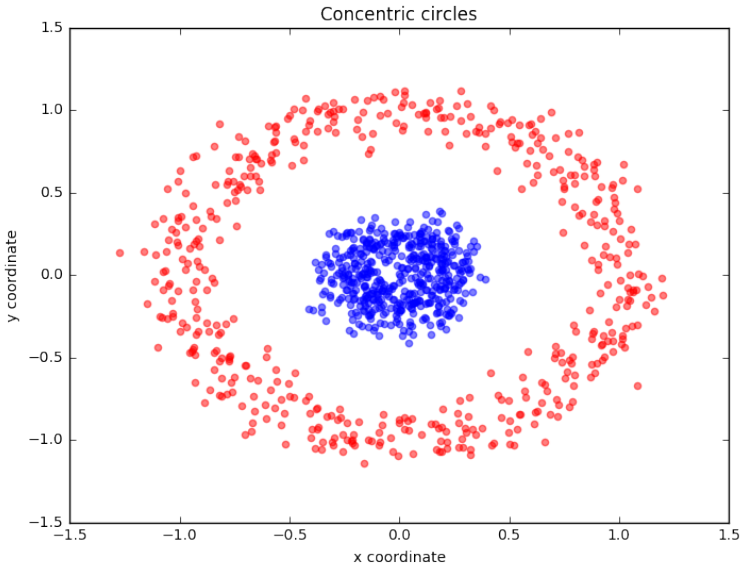

Fig. 1.3.2: Original dataset.

1.4 Motivation

Several clustering algorithms have been used in different applications such as image segmentation, data mining and others. k-means, for instance, is one of the most widely used clustering algorithms. It clusters data by minimizing the total within cluster variances based on Euclidean distances between pairs of data points. However, with increase in the dimensionality of data, Euclidean distances between pairs of data points approach a constant value. As a result, clustering algorithms based on

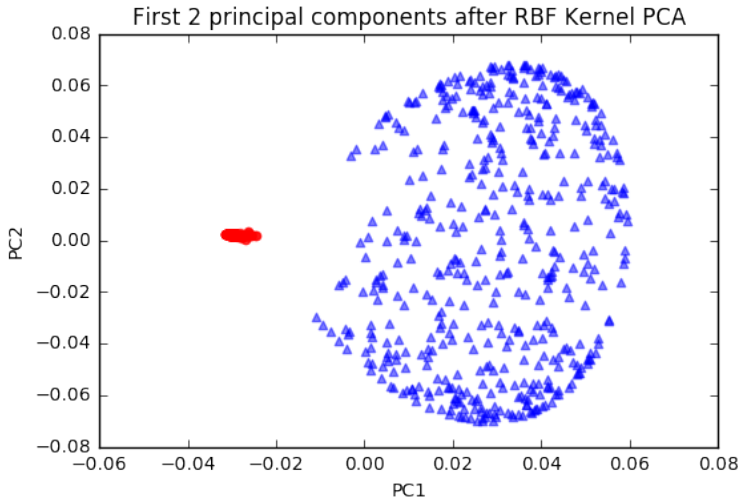

Fig. 1.3.3: KPCA transformation using RBF kernel and $\gamma = 10.0$ as parameter.

the Euclidean norm show limited performance in case of high-dimensional data in which points are related via complex, hidden relationships not captured by Euclidean distances.

In order to overcome this problem, high-dimensional data is transformed onto lower dimensional data, which then can be clustered based on the Euclidean norm or the L_2 metric. Dimensionality reduction can be performed either based on feature selection or feature extraction. In feature selection, only a small subset of the original features is retained by discarding other redundant or less important features at the cost of losing some information. In case of feature extraction, a smaller number of features are generated (extracted) from the original features by preserving the distinguishing characteristics of the original high-dimensional data.

Most of the approaches for dimensionality reduction require many parameters, or even hyper-parameters, which have to be tuned in tandem with the final result, namely, the quality of the clustering. If k-means is used for clustering in combination with different dimensionality reduction techniques, then, the entire process suffers from the limitations of k-means such as non-repeatability of clustering results, high dependency on initialization of cluster means and other factors. Also, the simplest forms of finding k centers using the Euclidean distance is known to be NP-complete [2] [3], when dealing with more than two clusters in two (or higher) dimensional data

[7]. Thus, k-means does not guarantee optimality and may converge locally while the clustering task is performed.

1.5 Proposed Method

In this thesis, we propose an approach for efficient clustering of high-dimensional, complex data. The proposed clustering approach has two stages that involve dimensionality reduction via kPCA, followed by optimal, one-dimensional clustering using a dynamic programming algorithm that runs in $O(n^2k)$ time. The use of optimal, one-dimensional clustering avoids the drawbacks associated with k-means clustering. Also, using kernels such as RBF in k-PCA eliminates the need to tune multiple parameters or even hyper-parameters, thereby allowing to search a large range of values for a single parameter. Finally, the quality of clustering is measured using the Silhouette score.[8] The advantages of the proposed approach are demonstrated in comparison with standard k-means clustering in terms of optimality and repeatability.

1.5.1 Contributions

The main contributions of this thesis can be summarized as follows:

- Proposed an efficient clustering approach which uses non linear dimensionality reduction and optimal one-dimensional clustering achieving polynomial-time complexity.
- Combined the power of Kernel Principal Component Analysis and one-dimensional clustering using the dynamic programming approach.
- Demonstrated the advantages of the proposed one-dimensional clustering approach over standard k-means clustering using real world and synthetic datasets in terms of time complexity, optimality and repeatability of clustering results.

- Implemented the proposed approach in Python, and developed an open-source project that is available at `https://github.com/Nachiket-Bhide/kPCA-and-Optimal_One_`

Dimensional_Clustering

References

[1] "A comparison of PCA , KPCA and ICA for dimentionality reduction". In: 2012.

[2] Daniel Aloise et al. "NP hardness of Euclidean sum-of-squares clustering". In: *Machine Learning*. Vol. 75(2). Springer, 2009, pp. 245–248. DOI: 10.1007/s10994-009-5103-0.

[3] Sanjoy Dasgupta and Yoav Freund. "Random Projection Trees for Vector Quantization". In: *IEEE Transactions on Information Theory* 55.7 (2009), pp. 3229–3242. DOI: 10.1109/TIT.2009.2021326.

[4] S Haykin. *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, New Jersey, United States: Prentice-Hall, 1999.

[5] I Jolliffe. *Principal Component Analysis*. New York, United States: Springer-Verlag, 1986. DOI: 10.1007/978-1-4757-1904-8.

[6] Stuart Lloyd. "Least squares quantization in PCM". In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: 10.1109/TIT.1982.1056489.

[7] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. "The planar k-means problem is NP-hard". In: *Theoretical Computer Science* 442 (2012). Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009), pp. 13–21. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2010.05.034.

[8] Peter Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: 10.1016/0377-0427(87)90125-7. URL: http://www.sciencedirect.com/science/article/pii/0377042787901257.

[9] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis". In: *International Conference of Artificial Neural Networks*. Berlin, Germany: Springer, 1997, pp. 583–588. DOI: 10.1007/BFb0020217.

CHAPTER 2

Dimensionality Reduction and Optimal One-dimensional Thresholding

2.1 Introduction

Clustering is an unsupervised machine learning technique in which unlabelled data is partitioned into different groups by identifying the hidden commonalities in the data. Given a data set of n points in dimension d, the main goal of a clustering algorithm is to partition the data into k clusters such that data points in one cluster are more similar to each other compared to data points outside of that cluster. Several clustering algorithms have been used in different applications such as image segmentation, data mining and others. k-Means, for instance, is one of the most widely used clustering algorithms. It clusters the data by minimizing the total within-cluster variances based on the Euclidean distance between the pairs of data points. However, with increase in the dimensionality of data, Euclidean distances between pairs of data points approach a constant value. As a result, clustering algorithms based on the Euclidean norm show limited performance in case of high-dimensional data wherein Euclidean distances do not necessarily represent the similarity between data points. Examples of these are vertices or edges in a graph, bags of words, sets of documents, and many others.

In order to overcome this problem, high-dimensional data is transformed onto

lower dimensional data, which then can be clustered based on Euclidean norm or L_p metrics. Dimensionality reduction can be performed either based on feature selection or feature extraction. In feature selection, only a small subset of original features is retained by discarding other redundant or less important features at the cost of losing some information. In case of feature extraction, smaller number of features are generated (extracted) from the original features by preserving the distinguishing characteristics of original high-dimensional data.

High dimensional data can be visualized as a set of data points lying on an embedded non-linear manifold within the higher-dimensional space. Techniques such as Kernel Principal Component Analysis (kPCA), Spectral Clustering [11], Autoencoders, Self-organizing maps [14], isometric mapping (Isomap) [17] and other metric and non-metric techniques have been used to perform non-linear dimensionality reduction. In [2], E. Banjamali et al. have proposed a fast spectral clustering approach based on autoencoders and landmarks which reduces the time complexity of traditional spectral clustering. In [16], S.Tasoulis et al. have proposed an approach which uses Isomap to recursively embed subsets of high-dimensional data in one dimension followed by hierarchical clustering based on binary partitioning. In [12], P. Nousil et al. have proposed an approach for clustering high dimensional data wherein autoencoder is used for dimensionality reduction and the resulting low dimensional representation of the data is clustered using the k-means algorithm. Most of these approaches for dimensionality reduction require many parameters, or even hyper-parameters, which have to be tuned in tandem with the final result, namely the quality of the clustering. If k-means is used for clustering in combination with different dimensionality reduction techniques, then, the entire process suffers from the limitations of k-means such as non-repeatability of clustering results, high dependency on initialization of cluster means and others. Also, the simplest forms of finding k centers is NP-complete in Euclidean space even if number of clusters (k) is two [1] [3] or when dealing with more than two clusters in two dimensional data [8]. Thus, k-means does not guarantee optimality and may converge locally during clustering.

In this thesis, we propose an approach for efficient clustering of high-dimensional,

complex data. The main goal is achieved in two stages that involve dimensionality reduction via kPCA, followed by optimal, one-dimensional clustering using dynamic programming in $O(n^2k)$ time. The use of optimal one-dimensional clustering using dynamic programming avoids the drawbacks associated with k-means based clustering. Also, using kernels such as Radial Basis Function (RBF) in k-PCA eliminates the need to tune multiple parameters or even hyper-parameters, thereby allowing to search a large range of values for a single parameter such as γ in RBF. Finally, the quality of clustering is measured using the Silhouette score. The advantages of the proposed approach are demonstrated in comparison with standard k-means clustering in terms of optimality and repeatability.

2.2 Proposed Method

Efficient clustering of high-dimensional data is achieved by the proposed approach, which combines two main stages involving dimensionality reduction followed by optimal one-dimensional clustering.

High-dimensional data can be visualized as a set of data points lying on a non-linear manifold in a high-dimensional space. In the first stage, kPCA is used to perform dimensionality reduction and transform high-dimensional data onto its one-dimensional representation. In the next stage, the output of the previous stage, i.e., the one-dimensional representation of the original data is then clustered using an optimal one-dimensional clustering algorithm in $O(n^2k)$ time, where n is the total number of data points and k is the number of clusters. These two stages are further discussed in detail in the following subsections.

2.2.1 Kernel Principal Component Analysis

PCA is a widely used technique that performs dimensionality reduction by obtaining orthogonal projections of high-dimensional data onto a lower-dimensional linear space, such that the dispersion of the projected data in terms of the eigenvectors of the within-class scatter is maximized. Consider a high-dimensional data set

$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ where $\mathbf{x_i}$ represents a single data point in the d-dimensional Euclidean space. PCA is a linear projection method from the d-dimensional input space to the p-dimensional output space ($p <<< d$) by solving eignevalue and eigenvector problem as follows:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}, \tag{1}$$

where $\mathbf{C}$ is the covariance matrix of the centered data:

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i}).(\mathbf{x_i})^T, \tag{2}$$

where λ and $\mathbf{v}$ are the eignevalues and eigenvectors of $\mathbf{C}$, respectively. Let $W = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_p\}$ be the matrix of p corresponding to the largest eigenvectors stacked in columns. The principal components $\mathbf{Y}$ are defined as:

$$\mathbf{Y} = \mathbf{W}^T\mathbf{X} \tag{3}$$

The new principal axes capture the maximum variance, such that data projected on the new axes are uncorrelated. However, classical PCA does not take into account the non-linear relationships among high-dimensional input data points [6]. To overcome this problem, kPCA introduced in [15] is widely used to extract non-linear features.

Non-linearity is introduced by mapping data from the input space R^N to a feature space F. As per Cover's theorem, nonlinear data structure in the input space is more likely to be linear after high-dimensional nonlinear mapping [5]. In kPCA, a nonlinear kernel function is used instead of the standard dot product, which implicitly performs PCA in the high-dimensional space F. Therefore, kPCA is able to produce features that capture nonlinear structure in the data more efficiently than linear PCA.

The mapping function is given as follows:

$$\begin{aligned} \phi : R^N &\longrightarrow F \\ \mathbf{x_i} &\longrightarrow \phi(\mathbf{x_i}) \end{aligned} \tag{4}$$

The correlation Matrix in the feature space F is defined as follows:

$$\tilde{\mathbf{C}} = \frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{x_i})\phi(\mathbf{x_i})^T, \tag{5}$$

kPCA is based on solving the eigenvector problem in the transformed space:

$$\tilde{\mathbf{C}}\tilde{\mathbf{v}} = \tilde{\lambda}\tilde{\mathbf{v}}, \tag{6}$$

where $\tilde{\lambda}$ and $\tilde{\mathbf{v}}$ are eigenvalues and eigenvectors of $\tilde{\mathbf{C}}$ respectively. $\tilde{\mathbf{v}}$ lies in the span of $\phi(\mathbf{x_1}), \ldots, \phi(\mathbf{x_N})$ and is therefore a linear combination of $\phi(\mathbf{x_i})$ elements. It can be written as follows:

$$\tilde{\mathbf{v}} = \sum_{j=1}^{N} a_j\phi(\mathbf{x_j}) \tag{7}$$

The kernel function is defined as follows:

$$K(\mathbf{x_i}, \mathbf{x_j}) = (\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})) \tag{8}$$

To extract the principal components of any point $\mathbf{x}$, the image $\phi(\mathbf{x})$ of the point needs to be projected onto the P obtained eigenvectors. Eigenvectors $\{\tilde{\mathbf{y}}_\mathbf{1}, \ldots, \tilde{\mathbf{y}}_\mathbf{p}, \ldots, \tilde{\mathbf{y}}_\mathbf{P}\}$ are the non-linear principal components in feature space F. This can be expressed mathematically as follows:

$$\tilde{\mathbf{y}}_\mathbf{p} = \tilde{\mathbf{v}}_\mathbf{p}^\mathbf{T}\phi(\mathbf{x}) = \sum_{i=1}^{N} a_{pi}K(\mathbf{x_i}, \mathbf{x}) \tag{9}$$

2.2.2 Optimal One-dimensional Clustering

The output of the previous stage is a one-dimensional representation of the original high-dimensional data obtained by performing dimensionality reduction using kPCA. In this second stage, one-dimensional points are optimally clustered using the dynamic programming approach in $O(n^2k)$ time, where n represents the total number of data points to be clustered in k clusters[18]. Let $\{x_1, x_2, \ldots, x_n\}$ be an array of length $n+1$ representing the one-dimensional output of kPCA sorted in ascending order, where

the array points are indexed at positions starting from $[1, 2, \ldots, n]$. The main goal of clustering can be defined as the task of assigning the elements of a one-dimensional array into k clusters in such a way that the sum of squares of the within-cluster distances from each element to its corresponding cluster mean is minimized. More formally, the clustering objective function can be expressed as follows:

$$\Psi = min \sum_{j=1}^{k} \sum_{i=1}^{n} \omega_{ij}(x_i - \mu_j)^2, \tag{10}$$

where $\omega_{ij} = 1$ if x_i belongs to cluster j; otherwise, $\omega_{ij} = 0$ and μ_j is the mean of cluster j.

This problem can be formulated as the task of partitioning n data points into k clusters by using $k + 1$ thresholds [13]. The threshold set T is defined as an ordered set $T = \{t_0, t_1, ...t_k\}$, where t_0 is set by default at the starting position 0 in the array and t_k is set by default on the last data point x_n at position n in the array. Therefore, for partitioning the data into k clusters, the remaining $k-1$ thresholds need to be set. The data points in interval $(t_j, t_{j+1}]$ represent a single cluster, where $0 \leq j \leq k - 1$. In terms of thresholds, the objective function in (10) is expressed as follows:

$$\Psi_k(n) = min \sum_{j=1}^{k} \psi(t_{j-1}, t_j], 1 \leq t_1 < t_2 ... < t_{k-1} < n \tag{11}$$

where ψ is the cost of setting threshold and

$$\psi(t_{j-1}, t_j] = \sum_{i} (x_i - \mu)^2 \quad \forall x_i \in (t_{j-1}, t_j], \tag{12}$$

and μ is the mean of all points in the interval $(t_{j-1}, t_j]$.

For efficient clustering, a combination of optimal thresholds should be found in such a way that it minimizes the objective function (11). The most straightforward method to obtain optimal thresholds is the brute force approach in which the objective function (11) is evaluated for all possible combinations of thresholds and a threshold set T is selected for which the objective function is minimum. However, the major

drawback of brute force approach is that it is computationally expensive and requires an exhaustive search of all possible threshold values resulting in an exponential time complexity of $O(n^{k-1})$, where n is the number of data points and $k-1$ is the number of thresholds to be set.

Using dynamic programming, optimal thresholds can be found in $O(n^2k)$ time complexity. Based on this approach, the objective function as defined in (11) can be broken down into smaller sub-problems, where each sub-problem can be defined as the task of finding optimal thresholds that partition the array in interval $[1, l]$ into m clusters, where $l \leq n$ and $m \leq k$. The objective function of the sub-problem is given by:

$$\Psi_m^*(l) = min \sum_{j=1}^{m} \psi(t_{j-1}, t_j], 1 \leq t_1 < t_2 ... < t_{m-1} < l \tag{13}$$

By setting $m = k$ and $l = n$, Equation (13) minimizes the overall problem and is equal to Equation (11). Based on the objective function of the sub-problem, the following recurrence is obtained:

$$\Psi_m^*(l) = min \sum_{j=1}^{m-1} \psi(t_{j-1}, t_j + \psi(t_{m-1}, l] \tag{14}$$

$$\Psi_m^*(l) = min \quad \Psi_{m-1}^*(t_{m-1}) + \psi(t_{m-1}, l] \tag{15}$$

Based on the above recurrence relation, it is clear that if the thresholds of sub-problem $\Psi_{m-1}^*(t_{m-1})$ are not set to optimal so as to minimize the sub-problem objective function. Then, the overall objective function can never be minimized with non optimal sub-problem thresholds. The base case of the above recursive relation when $m = 1$ is given as follows:

$$\Psi_m^*(l) = \begin{cases} min\Psi_{m-1}^*(t_{m-1}) + \psi(t_{m-1}, l]), & \text{if } m > 1. \\ \psi(0, l] & \text{if } m = 1. \end{cases} \tag{16}$$

The task of finding the thresholds can be better understood with the help of the trellis structure as depicted in Figure 2.2.1.

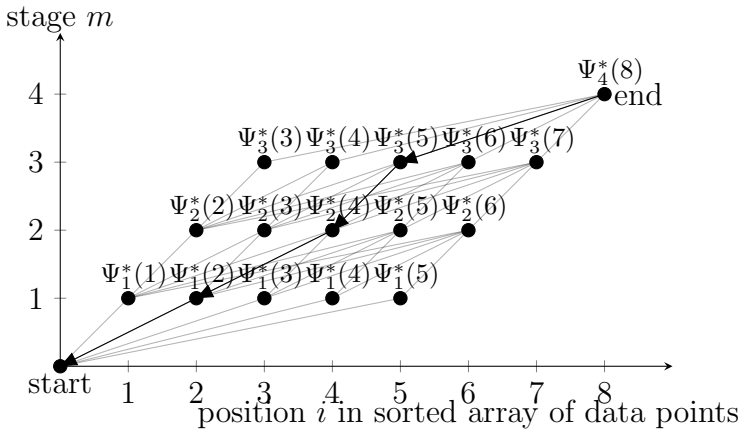

Fig. 2.2.1: An example of a trellis structure for partitioning $n = 8$ data points into $k = 4$ clusters.

The x axis represents the index number i in the sorted array of n data points, where $i = 1, \ldots, n$. The y axis represents stage m of the algorithm. For partitioning n data points into k clusters, threshold t_0 is set by default at index position $i = 0$ in the array, and threshold t_k is set by default on the last index position $i = n$ in the array. At each stage m for $m = 1, \ldots, k-1$, threshold t_m is set at index position i in the array. The goal is to find the path connecting *start* and *end* which minimizes the objective function as defined in Equation (16) for $m = k$ and $l = n$. Each node in the trellis structure represents the value of the objective function for each sub-problem $\Psi_m^*(l)$ as defined in Equation (13) and a back-pointer $ptr_m^*(l)$, which points to the position of the best node it comes from. At every node, the best node to come from and the resulting optimal cost are evaluated. The best path is stored in the node by setting a back-pointer and setting the value of node to the optimal cost accumulated so far.

Algorithm 1 describes the process of setting optimal thresholds using the dynamic programming approach. In the first stage, the trellis structure is initialized by computing the cost function for placing the thresholds at different indices in the array. Nodes in stages $1 < m < k$ are processed and the optimal path to each node is evaluated. Finally, backtracking is used to find all the thresholds. Algorithm 2 describes the function used to find the best path for each node. For each node in trellis struc-

ture, the function checks each possible node to come from and returns the optimal cost and the best position to set thresholds for that node [10].

Algorithm 2.2.1 Optimal 1D Clustering.

- - - - Stage 1- - - -
for $l \leftarrow 1$ to $n-k+1$ **do**
  $\Psi_1^*(l) \leftarrow l(0,l]$
  $ptr_1^*(l) \leftarrow 0$
end for
- - - - Stage 2- - - -
for $m \leftarrow 2$ to $k-1$ **do**
  for $l \leftarrow m$ to $n-k+m$ **do**
    $(\Psi_{min}, ptr) \leftarrow findoptimalpath(m,l)$
    $\Psi_m^*(l) \leftarrow \Psi_{min}$
    $ptr_m^*(l) \leftarrow ptr$
  end for
end for
- - - - Stage 3- - - -
$(\Psi_{min}, ptr) \leftarrow findoptimalpath(k,n)$
$\Psi_k^*(n) \leftarrow \Psi_{min}$
$ptr_k^*(n) \leftarrow ptr$
- - - - Backtracking- - - -
$l \leftarrow n$
for $m \leftarrow k$ to 2 **do**
  $t_{m-1} = l \leftarrow ptr_m^*(l)$
end for

2.2.3 Hungarian Algorithm

After each round of dimensionality reduction followed by one dimensional clustering, the data points belonging to the class having the highest cluster purity are removed. In order to achieve this, it is necessary to identify which cluster represents which class label. However, since data points belonging to different class labels might be spread across different clusters, it becomes necessary to assign class labels to resulting clusters in such a way that the total cost of assigning class labels to clusters is minimized. The objective of this linear assignment problem is to assign k class labels to k clusters such that the total cost of assignment is minimized. Mathematically, this can be

Algorithm 2.2.2 $findoptimalpath(m, l)$

```
Ψ_min ← ∞
for i ← m − 1 to l − 1 do
   Ψ_temp ← Ψ*_m − 1(i) + ψ(i, l]
   if Ψ_temp < Ψ_min then
      Ψ_min ← Ψ_temp
      ptr ← i
   end if
end for
return (Ψ_min, ptr)
```

expressed as [4]:

$$min \sum_{i=1}^{k} \sum_{j=1}^{k} c_{ij}, \tag{17}$$

where c_{ij} is the cost of assigning class label i to cluster j and it is defined as:

$$c_{ij} = \frac{|cluster(j)| - n_i^j}{\sum_{j=1}^{k} |cluster(j)|}, \tag{18}$$

where $|cluster(j)|$ is the number of data points in cluster j and n_i^j represents number of data points belonging to class i which are present in cluster j. In matrix form, the cost matrix can be represented as $C_{kxk} = [c_{ij}]$. The final assignment of class label i to cluster j which satisfies the objective function as defined in equation (17) is obtained using the Hungarian algorithm [7] which involves several row and column operations on cost matrix C.

2.3 Results and Discussion

For testing the proposed clustering approach, a real-life dataset of English letters and a synthetic dataset of randomly generated half-moons is used. High dimensional data is transformed into one dimensional representation, which is then clustered in polynomial time using the dynamic programming algorithm as explained in Section 2.2.

To evaluate the performance of proposed approach, the silhouette score is used.

The silhouette score is a metric used to measure how similar a data point is to its own cluster compared to other clusters. It is calculated using the mean intra-cluster distance, a, and the mean nearest-cluster distance, b, for each data point. The mean nearest cluster distance is the distance between a sample data point and the nearest cluster of which that sample data point is not a member. The distances used here is the Euclidean distance. The values of the silhouette score range from -1 to $+1$, where a value of $+1$ indicates that the clusters are nicely separated, whereas negative values indicate that the data points have been assigned to the wrong cluster.

$$Silhouette \quad Score = \frac{(b-a)}{max(a,b)} \tag{19}$$

2.3.1 Experimental Setup

For testing the proposed clustering approach, labelled datasets are used. The class to which each data point belongs to is known. In the first stage, a high-dimensional dataset is transformed into a one-dimensional representation using kPCA. The kernel function used in kPCA is the Radial Basis Function (RBF). The RBF kernel is defined as:

$$K(\mathbf{x_i}, \mathbf{x_j}) = exp(-\gamma||\mathbf{x_i} - \mathbf{x_j}||^2), \tag{20}$$

where $\mathbf{x_i}$ and $\mathbf{x_i}$ are input vectors, $\gamma = \sigma^{-2}$ is a parameter that depends on the variance, σ^2. The RBF projects vectors into an infinite-dimensional space. However, the RBF kernel represents similarity between a pair of vectors as a decaying function of the distance between the vectors without the need to perform an infinite-dimensional mapping. The closer the vectors are to each other, the smaller the value of $||\mathbf{x_i} - \mathbf{x_j}||$ is. This function is of the form of a bell-shaped curve. The parameter γ sets the width or spread of the bell-shaped curve. The larger the value of γ, the narrower the bell-shaped curve is. In the second stage, the one-dimensional representation of the data is clustered in polynomial time using the dynamic programming algorithm explained in Section 2.2.2. The class labels are assigned to the resulting clusters using the Hungarian algorithm as explained in Section 2.2.3. After each round of

dimensionality reduction followed by clustering, the data points belonging to the class which was the easiest to cluster are removed from the dataset. In the next round, the remaining high-dimensional data points are transformed to points in the one-dimensional space and then clustered. This process is repeated until the clustering of the data points belonging to the remaining two classes is performed.

2.3.2 Real-life Dataset

The real-life dataset used contains $20,000$ samples of the 26 letters of the English alphabet. Each sample is the 16 dimensional encoding, and there 26 different class labels corresponding to 26 letters of the English alphabet. The results are shown in the plot Figs. (2.3.1) and (2.3.2).

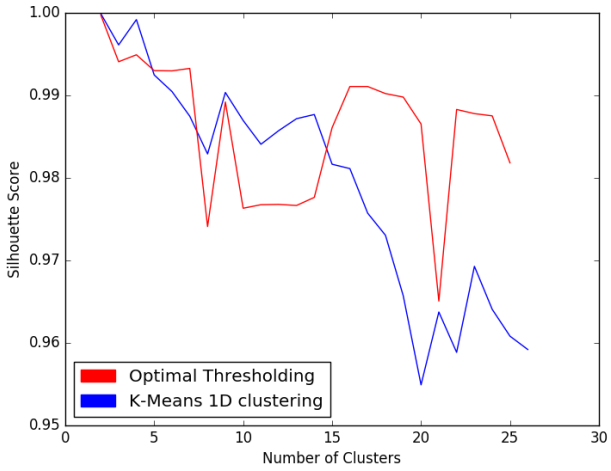

Fig. 2.3.1: k-means 1D vs Optimal One dimensional Clustering.

2.3.3 Synthetic Dataset

A synthetic dataset was generated by creating half moon pairs, each of which is displaced by a random distance and rotated by a random angle. There are 25 pairs of half-moons, thereby creating 50 clusters corresponding to 50 individual half-moons. The total size of the dataset is $10,000$. The sample dataset generated is shown in Fig. (2.3.3). The results are shown in Figs. (2.3.4) and (2.3.5).

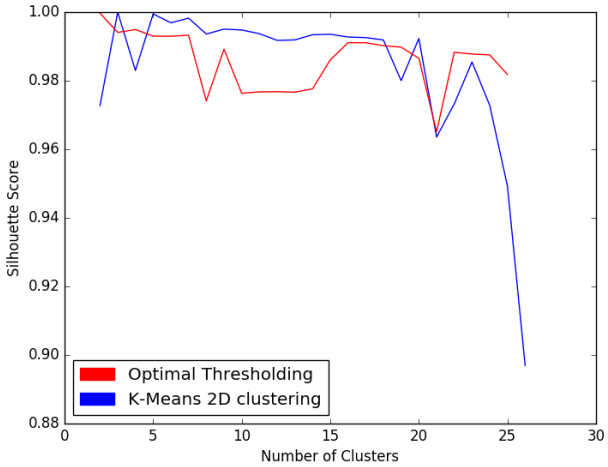

Fig. 2.3.2: k-means 2D vs Optimal One dimensional Clustering.

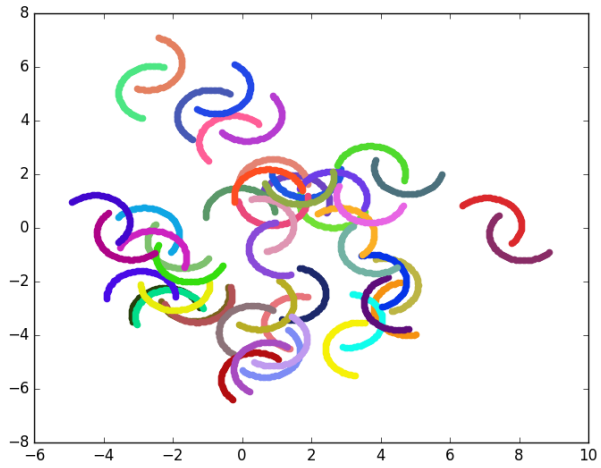

Fig. 2.3.3: Synthetic Dataset.

2.3.4 Discussion

The proposed clustering approach is tested on real-life and synthetic datasets. The performance of clustering is measured using the Silhouette score and the results are plotted as depicted in the previous sections. The advantage of the proposed approach is demonstrated by comparing its performance with that of the standard k-means clustering algorithm.

The plots show the Silhouette score of clusters obtained by using the proposed pipeline approach and by the k-means algorithm. It can be observed that as the number of clusters increases, the silhouette score of k-means decreases whereas the

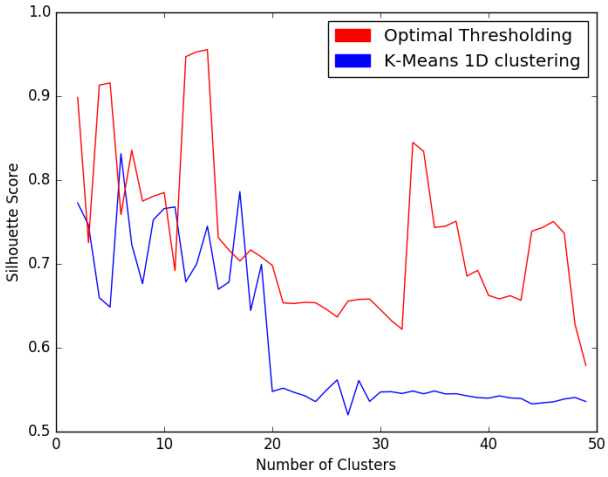

Fig. 2.3.4: k-means 1D vs Optimal One dimensional Clustering.

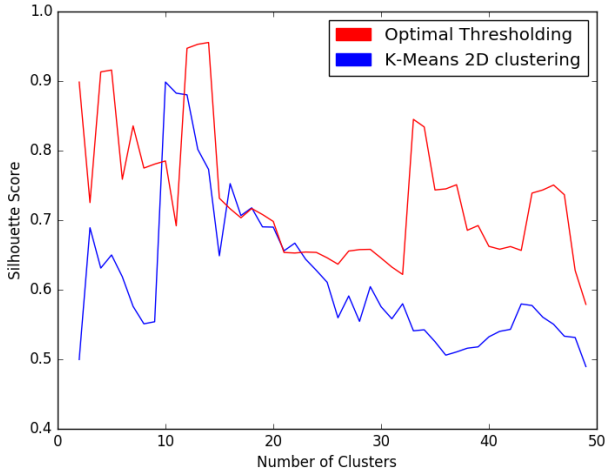

Fig. 2.3.5: k-means 2D vs Optimal One dimensional Clustering.

performance of the proposed optimal clustering approach increases. It has been shown that the k-center problem, in its simplest form of the problem solved by k-means, is NP-complete in Euclidean spaces even in dimension two. The time complexity of the k-means algorithm is $O(qknp)$, where q is the number of iterations, k is the number of clusters, n is size of the dataset and p is dimensionality [9]. k-Means on its own is heavily dependent on the initial cluster centers and the number of iterations, and hence it is neither repeatable nor optimal. In contrast to k-means, the proposed clustering approach does not depend on random initializations of cluster centers and hence its results are repeatable. Also, unlike k-means which may become stuck in

a local optimum, the proposed clustering approach guarantees optimal clustering in one dimension with $O(kn^2)$ time complexity.

References

[1] Daniel Aloise et al. "NP hardness of Euclidean sum-of-squares clustering". In: *Machine Learning*. Vol. 75(2). Springer, 2009, pp. 245–248. DOI: 10.1007/s10994-009-5103-0.

[2] Ershad Banijamali and Ali Ghodsi. "Fast Spectral Clustering Using Autoencoders and Landmarks". In: *International Conference Image Analysis and Recognition*. Springer-Cham, 2017, pp. -. DOI: 10.1007/978-3-319-59876-5-42.

[3] Sanjoy Dasgupta and Yoav Freund. "Random Projection Trees for Vector Quantization". In: *IEEE Transactions on Information Theory* 55.7 (2009), pp. 3229–3242. DOI: 10.1109/TIT.2009.2021326.

[4] Ketan Date and Rakesh Nagi. "GPU-accelerated Hungarian algorithms for the Linear Assignment Problem". In: *Parallel Computing* 57 (2016), pp. 52–72. ISSN: 0167-8191. DOI: 10.1016/j.parco.2016.05.012.

[5] S Haykin. *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, New Jersey, United States: Prentice-Hall, 1999.

[6] I Jolliffe. *Principal Component Analysis*. New York, United States: Springer-Verlag, 1986. DOI: 10.1007/978-1-4757-1904-8.

[7] Harold Kuhn. "The Hungarian method for the Assignment Problem". In: *Naval Research Logistics* Quart 2 (1955), pp. 83–97. DOI: 10.1002/nav.20056.

[8] Meena Mahajan, Prajakta Nimbhorkar, and Kasturi Varadarajan. "The planar k-means problem is NP-hard". In: *Theoretical Computer Science* 442 (2012). Special Issue on the Workshop on Algorithms and Computation (WALCOM 2009), pp. 13–21. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2010.05.034.

[9] Christopher Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. New York City, New York, United States: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[10] L. Martin and et al. "New Results on Efficient Optimal Multilevel Image Thresholding". In: *2006 International Conference on Image Processing*. Oct. 2006. DOI: 10.1109/ICIP.2006.312426.

[11] Andrew Ng, Michael Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*. MIT Press, 2001, pp. 849–856.

[12] Paraskevi Nousil and Anastasios Tefas. "Self-supervised autoencoders for clustering and classification". In: *Evolving Systems*. Vol. 11. Springer, 2018, pp. 453–466. DOI: 10.1007/s12530-018-9235-y.

[13] Luis Rueda. "An Efficient Algorithm for Optimal Multilevel Thresholding of Irregularly Sampled Histograms". In: *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*. SSPR amp; SPR '08. Orlando, Florida: Springer-Verlag, 2008, pp. 602–611. ISBN: 9783540896883. DOI: 10.1007/978-3-540-89689-0-64. URL: https://doi.org/10.1007/978-3-540-89689-0-64.

[14] Kaski Samuel. *Self-Organizing Maps*. Boston MA: Springer US, 2010, pp. 886–888. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8-746.

[15] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis". In: *International Conference of Artificial Neural Networks*. Berlin, Germany: Springer, 1997, pp. 583–588. DOI: 10.1007/BFb0020217.

[16] Sotiris Tasoulis, Nicos Pavlidis, and Teemu Roos. "Nonlinear dimensionality reduction for clustering". In: *Pattern Recognition* 107 (2020), p. 107508. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2020.107508.

[17] Joshua Tenenbaum, Vin Silva, and John Langford. "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500 (2000), pp. 2319–2323. ISSN: 0036-8075. DOI: 10.1126/science.290.5500.2319.

[18] Haizhou Wang and Mingzhou Song. "Ckmeans.1d.dp: Optimal k-means Clustering in One Dimension by Dynamic Programming". In: *The R Journal* 3/2 (2011).

CHAPTER 3

Conclusion and Future Work

3.1 Conclusion

In this thesis, we have proposed an approach for clustering high- dimensional data by combining the power of non-linear dimensionality reduction using kPCA and optimal one dimensional clustering using dynamic programming. The proposed clustering approach reduces the number of parameters in dimensionality reduction and thereby facilitates the task of searching a large range of values for a single kernel parameter. Also, by combining dimensionality reduction with optimal one-dimensional clustering, the drawbacks associated with k-means based clustering approaches are resolved.

3.1.1 Contributions

The main contributions of this thesis can be summarized as follows:

- Proposed an efficient clustering approach which uses non linear dimensionality reduction and optimal one-dimensional clustering achieving polynomial-time complexity.
- Combined the power of Kernel Principal Component Analysis and one-dimensional clustering using the dynamic programming approach.
- Demonstrated the advantages of the proposed one-dimensional clustering approach over standard k-means clustering using real world and synthetic datasets in terms of time complexity, optimality and repeatability of clustering results.

- Implemented the proposed approach in Python, and developed an open-source project that is available at `https://github.com/Nachiket-Bhide/kPCA-and-Optimal_One_Dimensional_Clustering`

3.2 Future Work

This thesis work can be further extended as follows:

- The proposed approach can be extended to other non-linear dimensionality reduction techniques apart from kPCA such as Isomaps, Laplacian Eigenmaps(LLE), or Locally Linear Embedding (LLE), just to mention a few.
- In case of parametric kernels, optimal values of the parameters need to be searched over a large range of possible values. To avoid this, non-parametric kernels can be used.
- The polynomial-time complexity of the one-dimensional thresholding based clustering could be further improved to a faster algorithm, yielding a wider range of parameters to consider, and subsequently improving the clustering.

VITA AUCTORIS

NAME:	Nachiket Bhide
PLACE OF BIRTH:	Nagpur, Maharashtra, India
EDUCATION:	B.E. Computer Engineering, Maharashtra Institute of Technology, Pune, Maharashtra, India, 2017
	M.Sc. Computer Science, University of Windsor, Windsor, Ontario, Canada, 2020