VOT 78096

GENERALIZED KERNEL METHODS FOR UNSUPERVISED LEARNING

DR. MOHD NOOR MD SAP
DR. SITI MARIYAM HJ. SHAMSUDDIN
DR. HARIHODIN SELAMAT
ABDUL MAJID AWAN
SHAFAATUNNUR BT. HASAN
MOJTABA KOHRAM

RESEARCH VOTE NO:
VOT 78096

Faculty of Computer Science and Information Systems
Universiti Teknologi Malaysia

2008

ABSTRACT

Unsupervised learning, mostly represented by data clustering methods, is an important machine learning technique. Data clustering analysis has been extensively applied to extract information from microarray gene expression data. However, finding good quality clusters in gene expression data is more challenging because of its peculiar characteristics such as non-linear separability, outliers, high-dimensionality, and diverse structures. Therefore, this study aims at combining kernel methods, capable of both handling the high dimensionality and discovering nonlinear relationships in the data, with the approximate reasoning offered by fuzzy approach. To this end, a robust Weighted Kernel Fuzzy C-Means incorporating local approximation (WKFCM) is presented. In WKFCM, fuzzy membership of each object is approximated from the memberships of its neighbouring objects. It brings in the synergy of partitioning and density based clustering approaches and provides a substantial improvement in the analysis of the data using unsupervised learning. Comparative analysis with K-means, hierarchical, fuzzy C-means and fuzzy self-organizing maps showed that, although different types of datasets are better partitioned by different algorithms, WKFCM displays the best overall performance, and has the ability to capture nonlinear relationships and non-globular clusters, and identify cluster outliers.

Keywords: Clustering; Kernel methods; Pattern recognition; microarray data analysis; gene expression data; Fuzzy C-means clustering (FCM)

ABSTRAK

Analisa pengelompokan data adalah suatu kelas yang besar dalam penggunaan pembelajaran tanpa penyeliaan. Ia telah meluas diaplikasikan untuk memperoleh informasi daripada susunan-mikro perwakilan data genetik. Walaubagaimanapun, mencari kualiti pengelompokan data yang baik adalah lebih mencabar kerana ia mempunyai karakter yang khusus seperti pemisahan tidak sekata, titik-luar, dimensi yang tinggi, dan mempunyai pelbagai struktur. Oleh itu, penyelidikan ini dijalankan adalah bertujuan untuk menyatukan kaedah *kernel* dimana mampu menggalas dimensi yang tinggi dan menemukan perhubungan tidak sekata dengan data, iaitu dengan mengaplikasikan anggaran munasabah yang ditawarkan oleh pendekatan kabur. Maka, pendekatan yg mantap iaitu *Weighted Kernel Fuzzy C-means (WKFCM)* yang menggabungkan anggaran setempat telah diperkenalkan. Keahlian kabur di dalam setiap objek *WKFCM* dianggarkan daripada keahlian objek jirannya. Ia membawa kepada kerjasama dalam pembahagian dan berdasarkan pendekatan kepadatan kelompok. Analisa Perbandingan dengan *K-Means, hierarchical, fuzzy C-Means,* dan *Fuzzy Self–Organizing Map* menunjukkan bahawa WKFCM tetap mempamerkan yang terbaik daripada keseluruhan pelaksanaan dan mempunyai kebolehan untuk mengenal pasti perhubungan tidak sekata, pengelompokan tidak global dan pengelompokan titik-luar walaupun pelbagai jenis data boleh diasingkan dengan algoritma yang lain.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

Unsupervised learning, mostly represented by data clustering methods, is an important machine learning technique. *Clustering* is a division of data into groups of similar objects. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is *unsupervised learning*, and the resulting system represents a data concept. From a practical perspective clustering plays an outstanding role in data mining applications such as scientific data exploration, information retrieval and text mining, spatial database applications, web analysis, marketing, medical diagnostics, computational biology, and many others. There are many approaches to data clustering that vary in their complexity and effectiveness, due to the wide number of applications that these algorithms have. While there has been a large amount of research into the task of clustering, currently popular clustering methods often fail to find high-quality clusters. Clustering has received a renewed attention with the advent of nonlinear clustering methods based on kernels as it provides a common means of identifying structure in complex data.

1.2 Background and General Problem Statement

Over the last decade, estimation and learning methods utilizing positive definite or Mercer kernels have become rather popular, particularly in machine learning. Since these methods have a stronger mathematical slant than earlier machine learning methods (e.g., neural networks), the statistics and mathematics communities have also significant interest in these methods [1]. Among these methods, Support Vector Machines (SVM) is being widely applied in the machine learning community since it often shows better performance than other learning algorithms. A distinctive feature of SVM is the use of Mercer kernels [2] to perform the inner product (kernel trick). The great success of SVM has led to the development of a new branch of machine learning, *Kernel Methods*, i.e. the algorithms that use the kernel trick. The kernel methods are among the most researched subjects within machine learning community in recent years and have been widely applied to pattern recognition and function approximation. Two of the typical examples are support vector machines (SVM) [2, 3], and kernel principal component analysis [4].

The fundamental idea of the kernel methods is to first transform the original low-dimensional inner-product input space into a higher dimensional feature space through some nonlinear mapping where complex nonlinear problems in the original low-dimensional space can more likely be linearly treated and solved in the transformed space. In the higher dimensional space, data points are spread out, and a linear separating hyperplane may be found. This concept is based on Cover's theorem on the separability of patterns. According to the Cover's theorem, an input space made up of nonlinearly separable patterns may be transformed into a feature space where the patterns are linearly separable with high probability, provided the transformation is nonlinear and the dimensionality of the feature space is high enough [5]. However, usually such mapping into high-dimensional feature space will undoubtedly lead to an exponential increase of computational time. Fortunately, adopting kernel functions to substitute an inner product in the original space, which exactly corresponds to mapping the space into higher-dimensional feature space, is a favorable option. Therefore, the inner product form leads us to applying the kernel methods to cluster complex data [6, 7].

The standard "sum-of-squares" (such as Euclidean distance measure) based methods of partitioning (such as *K*-means, FCM) have proved to be effective for datasets having ellipsoidal cluster structures [8]. A disadvantage to these methods is that clusters can only be separated by a hyperplane. If the separation boundaries between clusters are nonlinear, for instance non-Euclidean structures in the data such as nonspherical shape clusters, then these methods fail. An attractive approach to solving this problem is to adopt the strategy of nonlinearly transforming the data into a high-dimensional feature space and then performing the clustering within this feature space. Linear separators in the feature space correspond to nonlinear separators in the input space [4]. However, as the feature space may be of high and possibly infinite dimension, then directly working with the transformed variables is an unrealistic option. However, as mentioned above, it is unnecessary to work directly with the transformed variables. It is the inner-products between points which are used and these can be computed using a kernel function in the original data space [2, 4]. This observation provides for a tractable means of working in the possibly infinite feature spaces. While powerful kernel methods have been proposed for supervised classification and regression problems, the development of effective kernel method for clustering, aside from a few tentative solutions [4, 6, 7, 9], needs further investigation [9, 10].

1.3 Objective of the Study

To study the state-of-the-art approaches to non-linear system modeling concerning fundamental theoretical aspects, design of efficient and reliable algorithms.

1.4 Scope of the Study

The scope of the study is as follows:

- This study focuses on the issue of clustering especially for microarray gene expression data analysis
- Mainly kernel-based methods have been used in this study
- Experimentation has been conducted on publicly available standard, real benchmark datasets.

1.5 Significance and Contribution of the Study

Clustering is a very useful tool for effective data analysis and has a wide range of applications. While a large number of clustering techniques have been developed in statistics, pattern recognition, data mining, and other fields, significant challenges still remain. Most of the clustering challenges, particularly those related to quality rather than computational resources, are the same challenges that existed years ago: how to find clusters with differing sizes, shapes and densities, how to handle noise and outliers. This study has come up with a new clustering algorithm, using kernel-based methods for effective and efficient data analysis by exploring structures in the data. The proposed clustering algorithm incorporates local neighborhood information for making more efficient with respect to noise and outliers. The algorithm has been successfully tested on simulated and benchmark datasets (iris data, microarray gene expression data).

1.6 Research Methodology

The exploration of complex datasets, for which no or very little information about the underlying distribution is available, fundamentally relies on the identification of ‘natural’ group structures in the data, a task which may be tackled

using clustering techniques. A cluster analysis can be seen as a three step process as outlined in Figure 1.1 [11]. The same methodology is adopted in this study.

The first step involves a number of data transformations including feature selection, normalization and the choice of a distance function, to ensure that related data items cluster together in the data space. When the data set is a set of vectors, as is the case with datasets considered in this study, it is often effective to linearly scale each attribute to zero mean and unit variance, and then apply the Gaussian radial basis function kernel or polynomial kernel [12]. The main advantage of this normalization is to avoid attributes in larger numeric ranges dominating those in smaller ranges. More advanced methods for kernel normalization are described in [13].

The second step consists of the selection, parameterization and application of one or several clustering methods. The resulting partitionings are evaluated in the third step using cluster-validation techniques. Cluster-validation techniques have the potential to provide an analytical assessment of the amount and type of structure captured by a partitioning, and should therefore be a key tool in the interpretation of clustering results [11].

The procedure of evaluating clustering results is known as *cluster validity.* Cluster validity methods may assist users in choosing clustering results independently from the clustering algorithms, the parameters and the number of clusters. In general there are three approaches to cluster validity: *external, internal* and *relative* criteria. For some datasets in our experiments reported here, we have class labels so external criteria are used to evaluate clustering results. The data for which class labels are not available, internal validity criteria are used. Please, see Chapter 2, Section 4.1 for more detail.

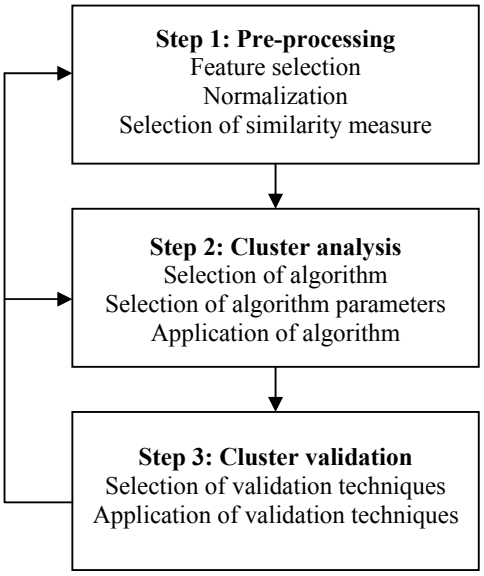


Figure 1.1 The three main steps involved in a cluster analysis: Preprocessing, cluster analysis, cluster validation

CHAPTER 2

A WEIGHTED FUZZY KERNEL BASED METHOD INCORPORATING LOCAL APPROXIMATION FOR CLUSTERING MICROARRAY DATA

Abstract

Data clustering analysis has been extensively applied to extract information from microarray gene expression data. However, finding good quality clusters in gene expression data is more challenging because of its peculiar characteristics such as non-linear separability, outliers, high-dimensionality, and diverse structures. Therefore, this study aims at combining kernel methods, capable of both handling the high dimensionality and discovering nonlinear relationships in the data, with the approximate reasoning offered by fuzzy approach. To this end, a robust Weighted Kernel Fuzzy C-Means incorporating local approximation (WKFCM) is presented. In WKFCM, fuzzy membership of each object is approximated from the memberships of its neighboring objects. It brings in the synergy of partitioning and density based clustering approaches and provides a substantial improvement in the analysis of the data. Comparative analysis with K-means, hierarchical, fuzzy C-means and fuzzy self-organizing maps showed that, although different types of datasets are better partitioned by different algorithms, WKFCM displays the best overall performance, and has the ability to capture nonlinear relationships and non-globular clusters, and identify cluster outliers.

Keywords: Clustering; Kernel methods; Pattern recognition; microarray data analysis; gene expression data; Fuzzy C-means clustering (FCM)

1. Introduction

The task of clustering genes into functionally-similar clusters using expression data rests on the assumption that genes of similar function share similar expression profiles across various experimental conditions. Clustering algorithms have proved useful to help group together genes with similar functions based on gene expression profiles under various conditions or across different tissue samples [14-17]. Such partitioning can facilitate data visualization and interpretation, and it can be exploited to gain insight into the transcriptional regulation networks underlying a biological process of interest. By expanding functional families of genes with known function together with poorly characterized or novel genes may help understand the functions of many genes which are not explored yet.

Since the work of Eisen *et al*. [17] clustering methods have become a key step in microarray data analysis. Various clustering algorithms have been applied in the cluster analysis of genes, including HAC (hierarchical agglomerative clustering) [17], SOM (self-organizing maps) [18], CLIFF (Clustering via Iterative Feature Filtering) [19], and algorithms based on mixture models [20], neural networks [21], simulated annealing [22], and PCA (principle components analysis) [23]. There are also many works in co-clustering gene expression matrix, i.e., clustering genes and samples at the same time [24, 25].

However, microarray datasets tend to have very diverse structures due to the complex nature of biological systems. Because of this, none of the existing clustering algorithms perform significantly better than the others when tested across various datasets [11, 14, 16, 26, 27]. Popular algorithms, such as *K*-Means, hierarchical clustering and Self-Organizing Maps (SOM) [28], typically perform clustering on the basis of pairwise distances between genes. Consequently they may fail to reveal nonlinear relationships between gene expression profiles, and be unable to correctly represent a dataset with nonlinear structures [29]. Over the last few years, more sophisticated clustering approaches have been developed for microarray data clustering, such as CLIFF [19], co-clustering [24] and GenClust [26]. Though in some cases they perform better than the standard methods, none of them proved consistently better across different datasets [11]. Anyway, HAC remains the most

widely used clustering algorithm and has become a de facto standard for visualization of expression data, although it has been described to suffer from a number of limitations mostly deriving from the local decision making scheme for constructing clusters that joins the two closest genes or clusters without considering the data as a whole, and it is likely to be a poor choice for further analysis of the resulting clusters [16, 18, 30, 31]. But genes on any given array are not isolated entities: the expression level of a specific gene should affect, or share information with, its biological neighbors. It suggests that Microarray datasets represent the collective behavior of a population best studied jointly; and many current statistical techniques ignore this [32]. In addition, handling of outliers in microarray data is extremely important as one outlier can yield misleading results [14].

More recently, fuzzy clustering approaches have been considered because they may assign one gene to multiple clusters (fuzzy assignment), which may allow capturing genes involved in multiple biological processes. Fuzzy C-Means (FCM) associates each object with every cluster based on the relative distances between the object and the cluster centroids [33, 34]. During the last few years, a number of variants of FCM have been proposed including a variant that incorporates PCA and hierarchical clustering [35], FuzzySOM [36], and Fuzzy J-Means that applies variable neighborhood searching to avoid local minima [37]. However, these FCM based clustering approaches lack the ability to capture non-linear relationships [29]. Some of the fuzzy clustering approaches are based on Gaussian Mixture Models (GMM) [20, 38], which assume the dataset to be generated by a mixture of Gaussian distributions with certain probability. But, the expression data do not always satisfy the basic Gaussian Mixture assumption even after carrying out various transformations aimed at improving the normality of the data distributions [20].

Keeping in view the above mentioned observations, the aim of this study is to propose a clustering algorithm combining good performance and robustness by exploiting kernel-based methods which offer strength to deal with complex data non-linearly separable in the input space and by incorporating fuzzy clustering approach, especially for the analysis of complex data with fuzzy structures such as microarray gene expression data. To this end, a robust Weighted Kernel Fuzzy C-Means incorporating local approximation (WKFCM) is presented. WKFCM integrates local

approximation based on the influence of the neighboring objects with the kernel fuzzy approach. It brings in the synergy of partitioning and density based clustering approaches and provides a substantial improvement in the analysis of the target data.

This paper is organized as follows. In the next section, it is briefly pointed out how kernel-based methods can be useful for clustering non-linearly separable and high-dimensional data. In section 3, the proposed algorithm–a Weighted Kernel Fuzzy C-Means incorporating local approximation (WKFCM)–is presented which can be useful for handling of non-linear separability, noise, and outliers in the data. Experimental settings, including evaluation measures, datasets and parameters used, are given in section 4. In section 5, comparative evaluation of WKFCM's performance on microarray data is given. Finally the paper concludes in section 6.

2. Kernel Methods and Clustering in Feature Space

Over the last decade, estimation and learning methods utilizing positive definite or Mercer kernels have become rather popular, particularly in machine learning. Since these methods have a stronger mathematical slant than earlier machine learning methods (e.g., neural networks), the statistics and mathematics communities have also significant interest in these methods [1]. Among these methods, Support Vector Machines (SVM) is being widely applied in the machine learning community since it often shows better performance than other learning algorithms. A distinctive feature of SVM is the use of Mercer kernels [2] to perform the inner product (kernel trick). The great success of SVM has led to the development of a new branch of machine learning, *Kernel Methods*, i.e. the algorithms that use the kernel trick. The kernel methods are among the most researched subjects within machine learning community in recent years and have been widely applied to pattern recognition and function approximation. Two of the typical examples are support vector machines (SVM) [2, 3], and kernel principal component analysis [4].

The fundamental idea of the kernel methods is to first transform the original low-dimensional inner-product input space into a higher dimensional feature space through some nonlinear mapping where complex nonlinear problems in the original low-dimensional space can more likely be linearly treated and solved in the transformed space. In the higher dimensional space, data points are spread out, and a linear separating hyperplane may be found. This concept is based on Cover's theorem on the separability of patterns. According to the Cover's theorem, an input space made up of nonlinearly separable patterns may be transformed into a feature space where the patterns are linearly separable with high probability, provided the transformation is nonlinear and the dimensionality of the feature space is high enough [5]. However, usually such mapping into high-dimensional feature space will undoubtedly lead to an exponential increase of computational time. Fortunately, adopting kernel functions to substitute an inner product in the original space, which exactly corresponds to mapping the space into higher-dimensional feature space, is a favorable option. Therefore, the inner product form leads us to applying the kernel methods to cluster complex data [6, 7].

Figure 1 illustrates that the two classes in input space may not be separated by a linear separating hyperplane. However, when the two classes are mapped by a nonlinear transformation function, a linear separating hyperplane can be found in the higher dimensional feature space. Let a nonlinear transformation function ϕ maps the data into a higher dimensional space. Suppose there exists a function κ, called a kernel function, such that,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j). \tag{1}$$

As already mentioned, a kernel function is substituted for the dot product of the transformed vectors, and the explicit form of the transformation function ϕ is not necessarily known. Further, the use of the kernel function is less computationally intensive. The formulation of the kernel function from the dot product is a special case of Mercer's theorem [13].

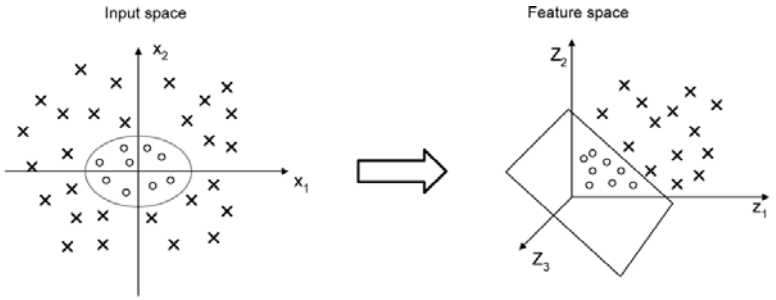


Figure 1 Mapping nonlinear data to a higher dimensional feature space where a linear separating hyperplane can be found. When mapped into a feature space via the non-linear map $\phi(\mathbf{x}) = (z_1, z_2, z_3) = \left([x]_1^2, [x]_2^2, \sqrt{2}[x]_1[x]_2 \right)$

The standard "sum-of-squares" (such as Euclidean distance measure) based methods of partitioning (such as *K*-means, FCM) have proved to be effective for datasets having ellipsoidal cluster structures [8]. A disadvantage to these methods is that clusters can only be separated by a hyperplane. If the separation boundaries between clusters are nonlinear, for instance non-Euclidean structures in the data such as nonspherical shape clusters, then these methods fail. An attractive approach to solving this problem is to adopt the strategy of nonlinearly transforming the data into a high-dimensional feature space and then performing the clustering within this feature space. To allow non-linear separators, kernel FCM (described in the next section) first uses a function ϕ to map data points to a higher-dimensional feature space, and then applies FCM in this feature space. Linear separators in the feature space correspond to nonlinear separators in the input space [4]. However, as the feature space may be of high and possibly infinite dimension, then directly working with the transformed variables is an unrealistic option. However, as mentioned above, it is unnecessary to work directly with the transformed variables. It is the inner-products between points which are used and these can be computed using a kernel function in the original data space [2, 4]. This observation provides for a tractable means of working in the possibly infinite feature spaces.

Examples of some well-known kernel functions are given in Table 1. We now develop the feature space FCM clustering method in the following section.

Table 1: Examples of popular kernel functions

Sigmoid Kernel	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \times \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \beta)$	γ and β are user defined values
Polynomial Kernel	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = < \mathbf{x}_i, \mathbf{x}_j >^d$	d is a positive *integer*
Gaussian Kernel (Radial Basis Function)	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2)$	σ is a user defined value

We use a small example to motivate the kernel idea. Suppose we want to cluster the 100 two-dimensional points in Figure 2(a) into 2 clusters such that points on the inner circle are in one cluster and the remaining points are in the other. None of the *K*-Means or the Fuzzy C-Means can generate the clustering that we want to see because they only discover clusters that are linearly separable.

Take the *K*-Means algorithm as an example. To decide whether $\mathbf{x}$ belongs to cluster V_1 or V_2, we compare distances $||\mathbf{x} - \mathbf{v}_1||$ and $||\mathbf{x} - \mathbf{v}_2||$. So all the points that are equally far from $\mathbf{v}_1$ and $\mathbf{v}_2$ satisfy the equation

$$||\mathbf{x} - \mathbf{v}_1|| = ||\mathbf{x} - \mathbf{v}_2||,$$

$$\text{i.e.,} \quad \mathrm{x}^T (\mathbf{v}_1 - \mathbf{v}_2) + (||\mathbf{v}_2||^2 - ||\mathbf{v}_1||^2) / 2 = 0,$$

which describes a hyperplane.

However, if we map the points into three-dimensional space using

$$\phi(\mathbf{x}) = [x_1^2, \sqrt{2}x_1x_2, x_2^2]^T \tag{2}$$

then points on different circles become linearly separable as shown in Figure 2(b) [39]. The *K*-Means algorithm should now be able to identify the two clusters.

Though mapping points to a higher dimensional space, called *kernel space* or *feature space,* enables a simple algorithm like the *K*-Means algorithm to handle non-linearly separable clusters, computing $\phi(\mathbf{x})$ can be slow especially when the kernel space has high dimensionality. However, if an algorithm only depends on the data through inner products, $\mathbf{x}^T\mathbf{z}$, in the original space, then after the mapping it will only depend on $\phi(\mathbf{x})^T\phi(\mathbf{z})$. Suppose we are given a *kernel function* $\kappa(\mathbf{x}, \mathbf{z})$, such that

$$\kappa(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^T\phi(\mathbf{z})$$

then we will not need to know ϕ or $\phi(\mathbf{x})$ to run the algorithm.

For the mapping function ϕ in (2), the corresponding kernel function is $\kappa(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T\mathbf{z})^2$, a degree 2 polynomial kernel, since

$$\begin{aligned}\kappa(\mathbf{x}, \mathbf{z}) &= \phi(\mathbf{x})^T\phi(\mathbf{z}) \\ &= x_1^2z_1^2 + \sqrt{2}x_1x_2\sqrt{2}z_1z_2 + x_2^2z_2^2 \\ &= (x_1z_1 + x_2z_2)^2 = (\mathbf{x}^T\mathbf{z})^2\end{aligned}$$

For a given set $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, matrix **K,** where $K_{st} = \kappa(\mathbf{x}_s, \mathbf{z}_t)$, $1 \le s, t \le n$, is called a *kernel matrix.* Since $\mathbf{K} = [\phi(\mathbf{x}_1),...,\phi(\mathbf{x}_n)]^T[\phi(\mathbf{x}_1),...,\phi(\mathbf{x}_n)]$ is the Gram matrix[1] of the images in the feature space; it is a symmetric, positive semidefinite matrix, and since it specifies the inner products between all pairs of points $\{\mathbf{x}\}_{i=1}^n$, it completely determines the relative positions of those points in the embedding space. On the other hand, if a given symmetric matrix **K** is positive semi-definite, we can compute the Cholesky decomposition

$$\mathbf{K} = \mathbf{R}^T\mathbf{R},$$

where **R** is an upper triangular matrix with non-negative diagonal. Then we can treat the columns of **R** as the images, thus **K** is a kernel matrix.

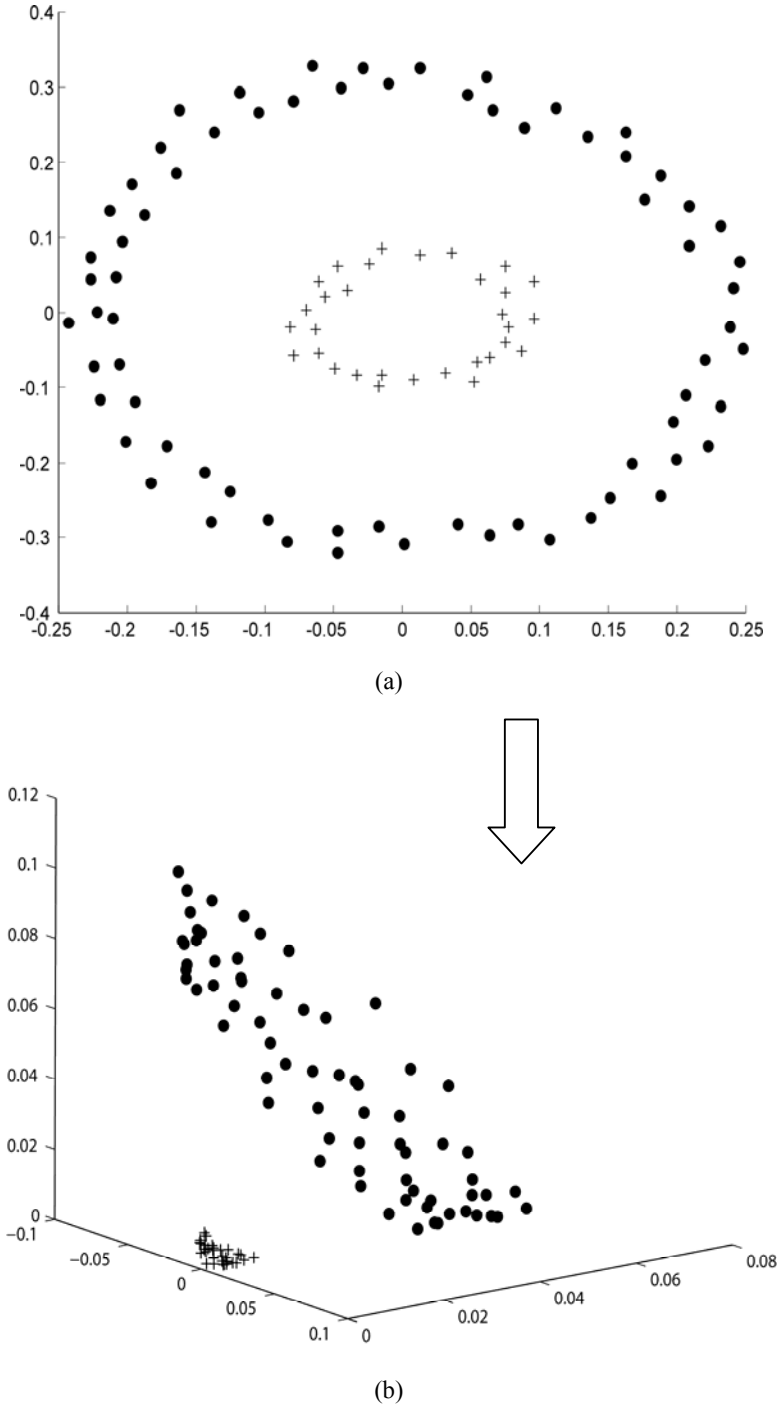


Figure 2 100 points distributed on two concentric circles:
(a) in the original space, (b) images of the points in the kernel space.

Table 2: Kernel matrix displays

K	1	2	. . .	*n*
1	$\kappa(\mathbf{x}_1, \mathbf{x}_1)$	$\kappa(\mathbf{x}_1, \mathbf{x}_2)$	. . .	$\kappa(\mathbf{x}_1, \mathbf{x}_n)$
2	$\kappa(\mathbf{x}_2, \mathbf{x}_1)$	$\kappa(\mathbf{x}_2, \mathbf{x}_2)$	. . .	$\kappa(\mathbf{x}_2, \mathbf{x}_n)$
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
n	$\kappa(\mathbf{x}_n, \mathbf{x}_1)$	$\kappa(\mathbf{x}_n, \mathbf{x}_2)$	. . .	$\kappa(\mathbf{x}_n, \mathbf{x}_n)$

where the symbol **K** in the top left corner indicates that the table represents a kernel matrix.

Definition: Gram matrix

Given a set of vectors, $S = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, the Gram matrix is defined as the $n \times n$ matrix **G** whose entries are $\mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. If we are using a kernel function κ to evaluate the inner products in a feature space with feature map ϕ, the associated Gram matrix has entries

$$\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \kappa(\mathbf{x}_i, \mathbf{x}_j)$$

In this case the matrix is often referred to as the *kernel matrix*. We will use a standard notation for displaying kernel matrices as shown in Table 2, where the symbol **K** in the top left corner indicates that the table represents a kernel matrix.

The Gram matrix plays an important role in some learning algorithms. The matrix is symmetric since $\mathbf{G}_{ij} = \mathbf{G}_{ji}$, that is $\mathbf{G}^T = \mathbf{G}$. Furthermore, it contains all the information needed to compute the pairwise distances within the dataset as shown above. This also reinforces the view that the kernel matrix is the central data type of kernel-based algorithms.

[1] The Gram matrix of **A** is $\mathbf{A}^T\mathbf{A}$

3. Weighted Kernel Fuzzy C-Means (WKFCM) incorporating local approximation

Clustering has received a significant amount of renewed attention with the advent of nonlinear clustering methods based on kernels as it provides a common means of identifying structure in complex data [6, 7, 9, 10, 40].

The aim of this study is to propose a clustering algorithm combining good performance and robustness by incorporating approaches of fuzzy clustering and kernel based methods, especially for analysis of complex data with fuzzy structures, such as microarray gene expression data. The algorithm approaches data clustering from a novel perspective. It is mainly based on two general assumptions: (a) clusters should be identified in the relatively dense parts of the dataset; (b) neighboring objects with similar features (expression profiles) must have similar cluster memberships so that the membership of one object is constrained or influenced by the memberships of its neighbors. Therefore, the membership of each single object (e.g., a gene or sample) is not only determined with respect to all other objects in the dataset or to some cluster centroids, but is also determined with respect to its neighboring objects. In addition to kernel space clustering, this approach also brings the notable advantage of capturing non-linear relationships, in a way similar to a nonlinear data dimensionality reduction approach called Locally Linear Embedding (LLE) [41, 42]. For LLE, the nonlinear relationships in a dataset are effectively captured by subdividing the general network of relationships across all objects into locally linear relationships between neighboring objects. Consequently, information about one object is approximated by the information obtained from its nearest neighbors. Inspired from this notion, we approached kernel fuzzy clustering based on neighborhood approximation to capture non-linear relationships in multidimensional data and to provide a substantial improvement in the analysis of the target data. The proposed clustering method, WKFCM, integrates the two above-mentioned key properties: (a) fuzzy membership assignment (gene-to-cluster relationship); (b) membership assignment under the influence of local approximation, where membership assignment of a gene also depends on the membership assignments of its neighboring genes (genes showing similar behavior).

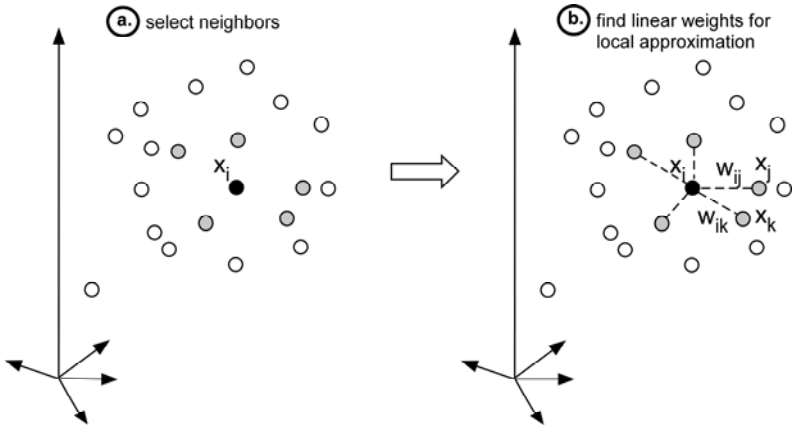


Figure 3 Steps for extracting local structure information: (a) Assign neighbors to each data point $\mathbf{x}_i$ by using the k nearest neighbors. (b) Compute the weights w_{ij} that best linearly approximate $\mathbf{x}_i$ from its neighbors, using the kernel similarity measures.

3.1 Extraction of Local Structure Information

Firstly the local structure information of the data is extracted. To this end, similarities between each pair of objects are calculated (a kernel function is used for measuring similarities, as described below), and the nearest neighbors are identified. The similarity measures between each object and its nearest neighbors are used to estimate the density around that object and to calculate a set of weights for local approximation in the next step. The set of densities forms a rough estimation of the distribution of the dataset, and the resulting values are also used in this step to identify possible cluster outliers.

The *K*-nearest neighbors (K_{NN}) for each gene are defined as k genes with the highest similarity according to a given similarity measure (kernel similarity measure). The weights defining how much each neighbor will contribute to the approximation of the membership of the object (say, object$_i$) are calculated as W_{ij}, as shown in Figure 3, with the following relation:

$$\sum_{j \in KNN(i)} w_{ij} = 1, \tag{3}$$

from the similarities s_{ij} between that gene (gene$_i$) and its nearest neighbors. The only requirement for a definition of weights is that, the neighbors that have higher similarities must get higher weights. The simplest one we use is:

$$w_{ij} = \frac{s_{ij}}{\sum_{j \in KNN(i)} s_{ij}} = \frac{\kappa(\mathbf{x}_i, \mathbf{x}_j)}{\sum_{j \in KNN(i)} \kappa(\mathbf{x}_i, \mathbf{x}_j)}. \quad (4)$$

In other words, the data to be fed to the main iterative procedure for clustering (described in the next subsection) becomes,

$$\tilde{\mathbf{x}}_i = \sum_{j \in KNN(i)} w_{ij} \mathbf{x}_j , \quad (5)$$

The distance measure is transformed into similarity measure using kernel based transformation to highlight relative proximities of the objects. As the elements of the kernel matrix represent similarities between the respective objects, following the above reasoning, the weights for individual objects can be defined as:

$$w_i = \frac{\sum_{j \in KNN(i)} \kappa(\mathbf{x}_i, \mathbf{x}_j)}{K_{NN}}, \quad (6)$$

where K_{NN} is the number of nearest neighbors.

The values of the weights for respective objects indicate the relative density around the objects or local density of the objects. The densely populated objects will get higher weights while the outliers and noise points will get lower weights. The first step is the extraction of local structure information and identification of cluster core objects (CCOs); in other words, starting cluster centroids or seed objects. In this step, the similarity (proximity) between each object and its *K*-nearest neighbors is used to calculate object density. Objects with the highest density among their neighbors are identified as CCOs and they serve as starting prototypes for the clusters, based on the fact that many other objects show similar behavior. In other words, CCOs are defined as individual objects having a particularly high number of neighbors. The number of clusters in the data can be estimated based on the number of CCOs. An example is shown in Figure 5 where two CCOs are identified in a simulated data consisting of two clusters. It is remarked here that higher is the

number of number of *K*-nearest neighbors (K_{NN}), the less number of CCOs will be identified, resulting in the less number of generated clusters.

To define possible cluster outliers, a density threshold can be applied so that objects with a density below the threshold are defined as possible outliers (objects with atypical behavior). In addition, this step adds features of the density based clustering approach to the partitioning based clustering approach. In a sense, this local approximation acts as a regularizer and biases the solution toward piecewise-homogeneous labeling. As it can be observed in Figures 6 and 7 that after applying local approximation, the boundary points are shifted towards their cluster centroids; it results in arrangement of clouds of points smoother at the boundaries. To define outliers, if the outliers are expected in the data, we used the following threshold on densities (or weights of individual objects, i.e., weights written with single subscript; whereas the weights written with double subscript represent interconnecting weights):

$$\rho_w = \overline{w} - 2\sigma_w \tag{7}$$

where $\overline{w}$ stands for mean density and σ_w stands for standard deviation of the densities.

This approach of incorporating local approximation brings in the following advantages: 1) It gives the estimation of the number of clusters present in the data by identifying cluster core objects (CCOs) which have higher density as compared to their neighboring objects; 2) the iterative procedure of the algorithm starts with the probable cluster centroids (CCOs); it results in fast convergence (less number of iterations) to a global solution; 3) by approximating data points based on the values of their nearest neighbors, the clusters of relevant points become even more compact, whereas the outliers or noise points are less affected (due to RBF kernel function), thus rendering them easy to get treated; it also helps in fast convergence of the algorithm (in less number of iterations) as the iterative procedure converges fast on compact and well separated data.

After application of these initial steps, the main iterative procedure for clustering is applied, as discussed in the next subsection.

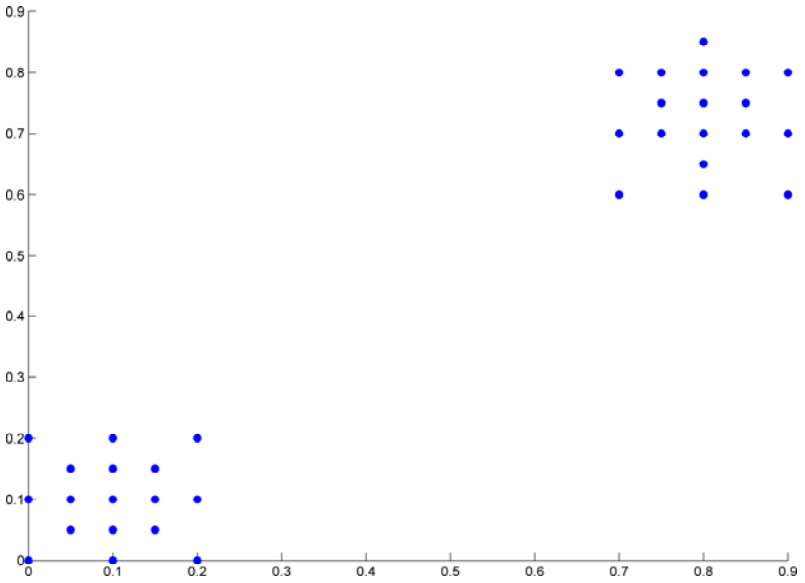


Figure 4 An example dataset (simulated Data-1) consisting of two clusters.

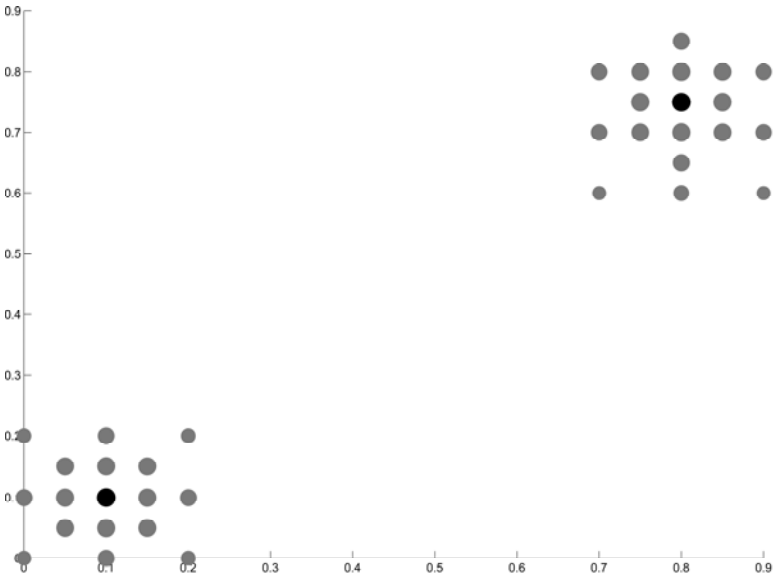


Figure 5 Data objects are used to calculate for each object a density value corresponding to the average similarity to its nearest neighbors using equation (6). In the Figure, the size of each point is proportional to density of the respective object in Figure 4; Cluster Core Objects (CCOs) are then identified as objects with maximum local density. The two black color objects define two CCOs. These CCOs serve as starting prototypes for the main iterative clustering procedure.

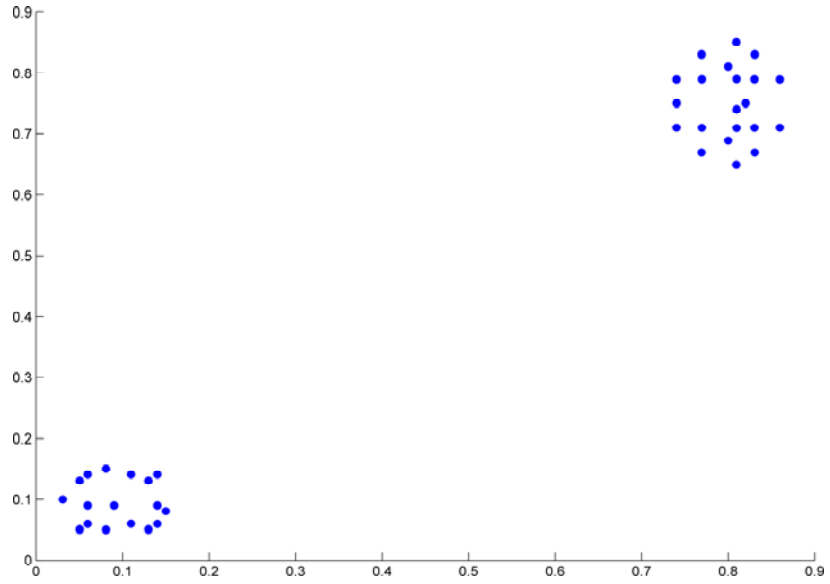
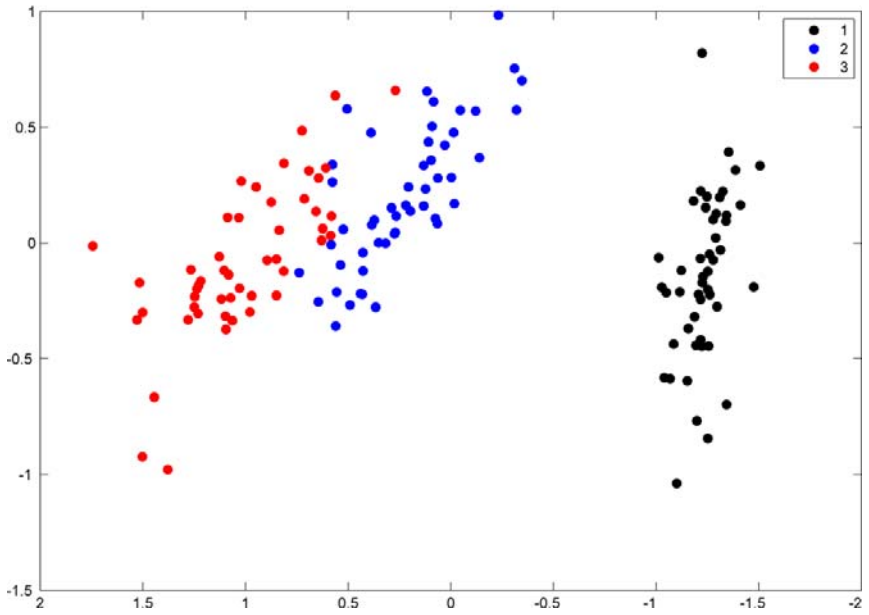
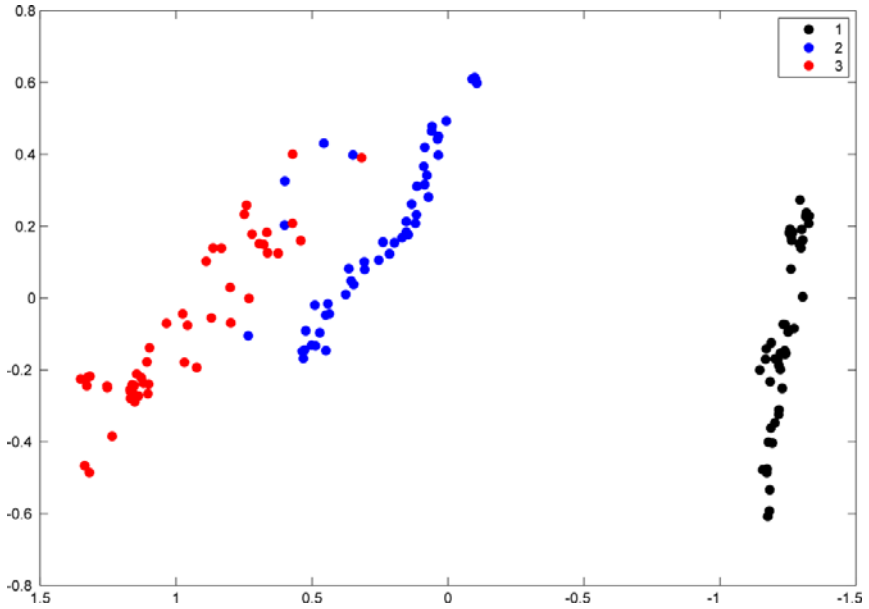


Figure 6 Applying local approximation. The simulated Data-1 (Figure 4) after applying local approximation using RBF with σ =0.5 and K_{NN} = 5. The clusters become compact and more separable on applying the approximation.



(a)



(b)

Figure 7 IRIS dataset. The data is projected along two major principal components. The three classes are represented by three different colors: (a) original dataset; (b) the dataset after applying neighborhood approximation using RBF with σ =0.7 and K_{NN} = 9. The clusters become compact and more separable on applying the approximation.

3.2 Approximation of Fuzzy Membership

Mathematically, the standard FCM objective function of partitioning a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^N$ (i.e., in N dimensional space) into c clusters, represented as $C = \{C_1, C_2, ..., C_c\}$, is given by

$$J_m = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m \left\| \mathbf{x}_i - \mathbf{v}_k \right\|^2, \tag{8}$$

where $\left\| \cdot \right\|$ stands for the Euclidean norm. Equivalently, (8) can, in an inner or scalar product form, be rewritten as

$$J_m = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m (\mathbf{x}_i^T \mathbf{x}_i - 2\mathbf{x}_i^T \mathbf{v}_k + \mathbf{v}_k^T \mathbf{v}_k), \tag{9}$$

where $V = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_c\}$ with $\mathbf{v}_k \in \mathbb{R}^N$ are the centroids or prototypes of the clusters $C_1, C_2, ..., C_c$; T denotes matrix transpose; the parameter m is a weighting exponent on each fuzzy membership and the array $\mathbf{U}=[u_{ik}]$ is a fuzzy partition matrix satisfying

$$U = \left\{ u_{ik} \in [0,1] \middle| \sum_{k=1}^{c} u_{ik} = 1, \forall i \text{ and } 0 < \sum_{i=1}^{n} u_{ik} < n, \forall k \right\}, \tag{10}$$

where u_{ik} denotes the membership degree of the ith pattern belonging to the kth cluster. Or,

$$\mathbf{U} = [u_{ik}] = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1c} \\ u_{21} & u_{22} & \cdots & u_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ u_{n1} & u_{n2} & \cdots & u_{nc} \end{bmatrix}.$$

And, $m \in [1, \infty)$

is a weighting exponent that controls the membership degree u_{ik} of each data point $\mathbf{x}_i$ to the cluster C_k. As $m \to 1$, J_1 produces a hard partition where $u_{ik} \in \{0,1\}$. As m approaches infinity, J_∞ produces a maximum fuzzy partition where $u_{ik} = 1/c$. This fuzzy c-means-type approach has advantages of differentiating how closely a gene belongs to each cluster [34] and of being robust to the noise in microarray data [43];

because it makes soft decisions in each iteration through the use of membership functions.

With the above formulations, we are now in a position to construct the kernelized version of the FCM algorithm and modify its objective function with the mapping ϕ as follows

$$J_m = \sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m \left\| \phi(\mathbf{x}_i) - \phi(\mathbf{v}_k) \right\|^2 . \tag{11}$$

Now, through the kernel substitution, we have

$$\left\| \phi(\mathbf{x}_i) - \phi(\mathbf{v}_k) \right\|^2 = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_i) - 2\phi(\mathbf{x}_i) \cdot \phi(\mathbf{v}_k) + \phi(\mathbf{v}_k) \cdot \phi(\mathbf{v}_k),$$

$$\left\| \phi(\mathbf{x}_i) - \phi(\mathbf{v}_k) \right\|^2 = \kappa(\mathbf{x}_i, \mathbf{x}_i) + \kappa(\mathbf{v}_k, \mathbf{v}_k) - 2\kappa(\mathbf{x}_i, \mathbf{v}_k), \tag{12}$$

where $\kappa(\mathbf{x}_i, \mathbf{x}_s) = K_{is} = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_s)$ is a user defined mercer kernel function, which can be used to represent a dot product in the high dimensional feature space. If the Gaussian radial basis function (RBF) is adopted, viz.

$$\kappa(\mathbf{x}_i, \mathbf{x}_s) = K_{is} = \exp\left(-\frac{\left\| \mathbf{x}_i - \mathbf{x}_s \right\|^2}{2\sigma^2} \right). \tag{13}$$

Then, in this case, $\kappa(\mathbf{x}_i, \mathbf{x}_i) = K_{ii} = 1$, so (11) can be simplified as

$$J_m = 2\sum_{k=1}^{c} \sum_{i=1}^{n} u_{ik}^m \left(1 - \kappa(\mathbf{x}_i, \mathbf{v}_k)\right) . \tag{14}$$

For optimization, the objective function J_m can be minimized if we take its first derivatives with respect to $\mathbf{v}_i$ and u_{ik}, and zero them, respectively, two necessary but not sufficient conditions for J_m to be at local minimum will be obtained as described below.

3.2.1 Cluster Prototype Updating

In order to minimize (14) with respect to v_k, we take the derivative of J_m with respect to v_k, and set the result to zero; so we have

$$\frac{\partial J_m}{\partial v_k} = -\frac{2}{\sigma^2}\sum_{i=1}^{n} u_{ik}^m \, \kappa(\mathbf{x}_i, \mathbf{v}_k)\,(\mathbf{x}_i - \mathbf{v}_k) = 0\,, \tag{15}$$

$$\mathbf{v}_k = \frac{\sum_{i=1}^{n} u_{ik}^m \; \kappa(\mathbf{x}_i, \mathbf{v}_k)\,\mathbf{x}_i}{\sum_{i=1}^{n} u_{ik}^m \; \kappa(\mathbf{x}_i, \mathbf{v}_k)},$$

or, $$\mathbf{v}_k = \frac{\sum_{i=1}^{n} u_{ik}^m \, \kappa(\mathbf{x}_i, \mathbf{v}_k)\,\mathbf{x}_i}{\sum_{i=1}^{n} u_{ik}^m \, \kappa(\mathbf{x}_i, \mathbf{v}_k)}. \tag{16}$$

3.2.2 Membership Evaluation

To optimize (14) with respect to u_{ik}, we can obtain the following Lagrange function without constraint,

$$J_m = 2\sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^m (1-\kappa(\mathbf{x}_i, \mathbf{v}_k)) - \sum_{i=1}^{n}\lambda\left(\sum_{k=1}^{c} u_{ik} - 1\right), \tag{17}$$

where λ is the Lagrange coefficient.

Rewrite (17) as follows:

$$J_m = 2\sum_{k=1}^{c}\sum_{i=1}^{n} u_{ik}^m (1-\kappa(\mathbf{x}_i, \mathbf{v}_k)) - \sum_{i=1}^{n}\lambda\left(\sum_{k=1}^{c} u_{ik} - 1\right), \tag{18}$$

where $(1-\kappa(\mathbf{x}_i, \mathbf{v}_k))$ is a weighted similarity measure in the kernel space.

Taking the derivative of J_m with respect to u_{ik} and setting the result to zero, we have, for $m > 1$,

$$\frac{\partial J_m}{\partial u_{ik}} = 2mu_{ik}^{m-1}(1-\kappa(\mathbf{x}_i, \mathbf{v}_k)) - \lambda = 0 . \quad (19)$$

Solving for u_{ik} we have

$$u_{ik} = \left(\frac{\lambda}{mw_i(1-\kappa(\mathbf{x}_i, \mathbf{v}_k))} \right)^{\frac{1}{m-1}} . \quad (20)$$

Considering the constraint $u_{ik} \in [0,1]$ and $\sum_{k=1}^{c} u_{ik} = 1,\ 1 \le i \le n,$ we have

$$\sum_{k=1}^{c} \left(\frac{\lambda}{mw_i(1-\kappa(\mathbf{x}_i, \mathbf{v}_k))} \right)^{\frac{1}{m-1}} = 1 , \quad (21)$$

or, $$\lambda = \frac{m}{\left(\sum_{k=1}^{c} \left(\frac{1}{m(1-\kappa(\mathbf{x}_i, \mathbf{v}_k))} \right)^{\frac{1}{m-1}} \right)^{m-1}} . \quad (22)$$

Substituting it into (20), the zero-gradient condition for the membership estimator can be re-written as

$$u_{ik} = \frac{(1-\kappa(\mathbf{x}_i, \mathbf{v}_k))^{\frac{1}{1-m}}}{\sum_{l=1}^{c} (1-\kappa(\mathbf{x}_i, \mathbf{v}_l))^{\frac{1}{1-m}}} . \quad (23)$$

This solution also satisfies the remaining constraints of Equation (10). Therefore, the cluster centroids and membership degrees in (16) and (23) are optimized in each iteration by minimizing the functional J_m.

3.3 Cluster Construction

On calculating sets of fuzzy membership values, either clusters can be defined based on a one-to-one gene-cluster assignment, or, one object can be assigned to more than one cluster if it has a reasonably high membership values for multiple clusters. Also, some objects may not be assigned to any cluster if they don't

have one dominant membership value. The objects not assigned to any cluster can be regarded as outliers or noise points. Such points can be screened out from the clusters.

WKFCM can be summarized in the following subsection.

3.4 Algorithm WKFCM

The algorithmic steps of WKFCM are as follows:

Algorithm Weighted Kernel Fuzzy *c*-Means (WKFCM)

WKFCM (**K**, [*c*], K_{NN})

Input: **K**: kernel matrix, *c*: number of clusters (optional), set ε > 0 to a very small value as a termination criterion, K_{NN} : number of nearest neighbors of a point,

Output: $v_1, ..., v_c$: partitioning of the points

1. Input the dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ with $\mathbf{x}_i \in \mathbb{R}^N$
2. For each object **x**, compute weights using equation (6),

$$w_i = \frac{\sum_{p \in KNN(i)} \kappa(\mathbf{x}_i, \mathbf{x}_p)}{K_{NN}}, \tag{6}$$

and find CCOs (cluster core objects) as initial cluster centroids (for *c* clusters: $\mathbf{v}_1, ..., \mathbf{v}_c$), and identify outliers, if any.

3. Approximate the data based on neighborhood information using the following relation:

$$\tilde{\mathbf{x}}_i = \sum_{p \in KNN(i)} w_{ip} \mathbf{x}_p$$

4. Set $r = 0$; initialize $\mathbf{U}^{(r)} = \left[u_{ik}^{(r)} \right]$ of $\mathbf{x}_i$ belonging to cluster C_k for $1 \leq k \leq c$, $1 \leq i \leq n$ such that $\sum_{k=1}^{c} u_{ik} = 1$.
5. Update the partition matrix using equation (23)

$$u_{ik}^{(r+1)} = \frac{(1-\kappa(\tilde{\mathbf{x}}_i, \mathbf{v}_k^{(r)}))^{\frac{1}{1-m}}}{\sum_{l=1}^{c}(1-\kappa(\tilde{\mathbf{x}}_i, \mathbf{v}_l^{(r)}))^{\frac{1}{1-m}}} \tag{23}$$

6. Update the centroids $V^{(r+1)} = \{\mathbf{v}_1^{(r+1)}, \mathbf{v}_2^{(r+1)}, \ldots, \mathbf{v}_c^{(r+1)}\}$ for $1 \le i \le c$ using equation (16)

$$\mathbf{v}_k^{(r+1)} = \frac{\sum_{i=1}^{n}(u_{ik}^{(r+1)})^m \kappa(\tilde{\mathbf{x}}_i, \mathbf{v}_k^{(r)})\tilde{\mathbf{x}}_i}{\sum_{i=1}^{n}(u_{ik}^{(r+1)})^m \kappa(\tilde{\mathbf{x}}_i, \mathbf{v}_k^{(r)})} \tag{16}$$

7. Stop if the following termination criterion is met:

$\left\|V^{(r+1)} - V^{(r)}\right\| < \varepsilon$

such as $\max\left(\left\|v_{kj}^{(r+1)} - v_{kj}^{(r)}\right\|, \text{ for } 1 \le k \le c \text{ and } 1 \le j \le N\right) \le 0.0001$

where $V = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\}$, or, the maximum number of iterations is reached. Otherwise, set r=r+1 and return to step 5.

4. Experimental Settings

Gene expression data are generated by DNA chips and other microarray techniques. The raw data produced by microarray often come along with noise, missing values and systematic variations [44]. Preprocessing, such as estimation of missing values [45], normalization [46, 47], is needed. After the above preprocessing steps, gene expression data can be represented as a real-valued matrix, in which the entry at row i and column j is the measured expression level of gene$_i$ under condition$_j$, as shown in Figure 8.

For comparative evaluation of WKFCM, the evaluation measures, datasets and parameters used are described in the following subsections.

	condition_1	…	condition_j	…	condition_N
gene_1	x_{11}	…	x_{1j}	…	x_{1N}
…	…	…	…	…	…
gene_i	x_{i1}	…	x_{ij}	…	x_{iN}
…	…	…	…	…	…
gene_n	x_{n1}	…	x_{nj}	…	x_{nN}

Figure 8 Gene expression data matrix

4.1 Evaluation Measures for Clustering

Evaluating clustering results is a tricky business. However, in situations where data points are already categorized (labelled), we can compare the clusters with the "true" class labels and calculate classification rate. To evaluate the goodness of the clustering produced by the algorithms on the test data without true class labels, two validity measures were used in this study: the Figures of Merit (FOM), and the Davies-Bouldin Index (DBI).

4.1.1 Classification accuracy

To compare the clustering results of different algorithms on the data for which class labels are known, classification accuracy or classification rate is defined as:

$$\text{Classification accuracy}\,(\%) = \frac{\text{Number of correctly classified points}}{\text{Total number of points}} \times 100$$

4.1.2 Figures of Merit

The FOM of Yeung et al. [48] estimates the predictive power of a clustering method based on the jackknife approach. The method measures the root mean square deviation in the left-out condition of the individual gene expression level relative to their within-cluster means. As each condition is used as the validation condition, it calculates the sum of FOMs over all the conditions. Meaningful clusters exhibit less variation in the remaining conditions than clusters formed by random. Thus, a lower value of FOM represents a well-clustered result, representing that a clustering method has high predictive power.

The use of Figures of Merit (FOMs) has been proposed by Yeung *et al*. [48, 49] to characterize the predictive power of different clustering algorithms. FOM is estimated by removing one experiment at a time from the dataset, clustering genes based on the remaining data, and then measuring the within-cluster similarity of the expression values in the left-out experiment. The principle is that correctly co-clustered genes should retain a similar expression level also in the left-out sample. The assumption (and limit) of this approach is that most samples have correlated gene expression profiles. The most commonly used FOM, referred to as "2-Norm FOM" [48], measures the within-cluster similarity as root mean square deviation from the cluster mean in the left-out condition. An aggregated FOM is obtained by summing up all the FOMs of all left-out experiments and is used to compare the performance of different clustering algorithms (the lower the FOM, the better the predictive power of a clustering algorithm). Since it is a rather novel measure, a formal definition is provided below.

For a given dataset, let R denotes the raw data matrix. Assume that R has dimension $n \times N$, i.e., each row corresponds to a gene and each column corresponds to an experimental condition. Assume that a clustering algorithm is given the raw matrix R with column e excluded. Assume also that, with that reduced dataset, the algorithm produces c clusters R_1, ..., R_c. Let $r_{(g,e)}$ be the expression level of gene g and $m_{(i,e)}$ be the average expression level of condition e for genes in cluster R_i. The 2-Norm FOM with respect to c clusters and condition e is defined as:

$$\mathrm{FOM}(e,c) = \sqrt{\frac{1}{n}\sum_{i=1}^{c}\sum_{g_k \in R_i}\left(r_{(g_k,e)} - m_{(i,e)}\right)^2}\,. \tag{24}$$

Notice that FOM(e,c) is essentially a root mean square deviation. The aggregate 2-Norm FOM for c clusters is then:

$$\mathrm{FOM}(c) = \sum_{e=1}^{N}\mathrm{FOM}(e,c)\,. \tag{25}$$

Both formulae (24) and (25) can be used to measure the predictive power of an algorithm. The first gives us more flexibility, since we can pick any condition, while the second gives us a total estimate over all conditions. Moreover, since the experimental studies conducted by Yeung *et al*. [48, 49] show that FOM(c) behaves as a decreasing function of c, an adjustment factor has been introduced to properly compare clustering solutions with different numbers of clusters. A theoretical analysis by Yeung *et al*. [48] provides the following adjustment factor:

$$\sqrt{\frac{n-c}{n}}\,. \tag{26}$$

When (24) is divided by (26), (24) and (25) are referred to as *adjusted* FOMs. We use the adjusted aggregate 2-Norm FOM for our experiments, and we refer to it simply as 2-Norm FOM.

4.1.3 Davies-Bouldin Index (DBI)

The Davies-Bouldin Index (DBI) aims at identifying sets of clusters that are compact and well separated [50]. Small values of DBI correspond to clusters that are compact, and whose centres are far away from each other. For any partition $W \leftrightarrow X : C_1 \cup C_2 \cup \ldots C_c$, where C_i represents the *ith* cluster of such partition, the DB index is defined as

$$DBI(W) = \frac{1}{c}\sum_{i=1}^{c} \max_{i \neq j} \left\{ \frac{\Delta(C_i) + \Delta(C_j)}{\delta(C_i, C_j)} \right\}, \tag{27}$$

here $\delta(C_i, C_j)$ defines the inter-cluster distance between the clusters C_i and C_j; $\Delta(C_i)$ represents the intracluster distance of cluster C_i , and c is the number of clusters of partition W.

Different methods may be used to calculate intercluster and intracluster distances [11]. Mathematical definitions of the intercluster and intracluster distances used in our experiments are given in the following subsections. For details, please see [11].

4.1.3.1 Intercluster Distances

Six intercluster distances may be used for the calculation of the Davies-Bouldin validity indices. The *single linkage* distance defines the closest distance between two samples belonging to two different clusters. The *complete linkage* distance represents the distance between the most remote samples belonging to two different clusters. The *average linkage* distance defines the average distance between all of the samples belonging to two different clusters. The *centroid linkage* distance reflects the distance between the centres of two clusters. The *average of centroids linkage* represents the distance between the centre of a cluster and all of samples belonging to a different cluster. *Hausdorff metrics* are based on the discovery of a maximal distance from samples of one cluster to the nearest sample of another cluster. In this study, *average linkage distance* is used which is defined below:

$$\delta(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x_1 \in C_1, x_2 \in C_2} d(x_1, x_2), \tag{28}$$

where C_1 and C_2 are clusters from partition W; $d(x_1, x_2)$ defines the distance between any two samples, x_1 and x_2, belonging to C_1 and C_2, respectively; $|C_1|$ and $|C_2|$ provide the number of samples included in clusters C_1 and C_2, respectively.

4.1.3.2 Intracluster Distances

Three intracluster distances may be used to calculate the Davies-Bouldin validity indices. The *complete diameter* distance represents the distance between the most remote samples belonging to the same cluster. The *average diameter* distance defines the average distance between all of the samples belonging to the same cluster. The *centroid diameter* distance reflects the double average distance between all of the samples and the cluster's centre. In this study, *average diameter distance* is used which is defined below:

$$\Delta(C_i) = \frac{1}{|C_i|(|C_i|-1)} \sum_{\substack{x_1, x_2 \in C_i \\ x_1 \neq x_2}} d(x_1, x_2), \tag{29}$$

where C_i is a cluster from partition W; $d(x_1, x_2)$ defines the distance between any two samples, x_1 and x_2, belonging to C_i; $|C_i|$ represents the number of samples included in cluster C_i.

4.2 Microarray Datasets and Analysis Parameters

To assess the performance of WKFCM and compare it with other popular algorithms, such as *K*-Means, Hierarchical clustering [35], Fuzzy C-means (FCM) [33], Fuzzy SOM (FSOM) [36], we used three different datasets: (i) Peripheral Blood Monocytes (PBM) dataset [26], (ii) yeast cell cycle (YCC) expression dataset [51], and (iii) hypoxia response (HR) dataset [15]. Further details on the datasets and parameters used are provided in the following subsections.

4.2.1 Peripheral Blood Monocytes (PBM) dataset

It is a reduced version of a Peripheral Blood Monocytes (PBM) dataset originally used by Hartuv *et al*. [52] to test their clustering algorithm. The dataset

contains 2329 cDNAs with a fingerprint of 139 oligos (performed with 139 different Oligonucleotide probes) derived from 18 genes. The spotted cDNAs derived from the same gene should display a similar profile of hybridization to the 139 probes and therefore be clustered together. Since FOM analysis is too time demanding, Di Gesu *et al*. [26] reduced the dataset (PBM) to contain 235 cDNAs. So, the dataset used for our experiments is also a 235×139 data matrix.

4.2.2 Yeast Cell Cycle (YCC) Data

This yeast cell cycle data is a part of the studies conducted by Spellman *et al*. [51]. The complete dataset contains about 6178 genes under 76 experimental conditions. The reduced yeast cell cycle (YCC) dataset is a subset of the original YCC dataset selected by Yeung *et al*. [48, 49] for FOM analysis and is composed of 698 genes under 72 experimental conditions. We also used the same dataset for our experiments.

4.2.3 Hypoxia Response (HR) Data

The hypoxia response (HR) dataset has been used by Chi *et al*. [15] to investigate cell type specificity and prognostic significance of gene expression programs in response to hypoxia in human cancers. The dataset was downloaded from Stanford Microarray Database with default filtering parameters provided by the web interface This way, a data subset of 6613 genes under 57 experimental conditions was obtained. After filtering out genes with more than 80% null values, we selected top 1000 genes with the highest expression variations.

4.2.4 Parameters

The following parameters were used for all the datasets: cosine correlation was used as a distance metric for all other methods except WKFCM; un-weighted pair-group average linkage for hierarchical clustering; 600 as a maximum number of iterations and ε=0.0001 as the converging criteria for all methods except hierarchical clustering. Clusters with a large range of cluster numbers were generated for the comparison. The fuzziness parameter m=1.2 was used for FCM, FSOM and WKFCM. In addition, for WKFCM, we used Gaussian RBF kernel with K_{NN}=4.

5. Evaluation of WKFCM

5.1 Experimentation on Simulated Data-2

The simulated dataset (Data-2) is a two-dimensional set formed by 111 points (86 points in one cluster, 19 points in the other cluster, 6 outliers). The Data-2 is shown in Figure 9. For comparison, we tried *K*-Means SOM [28] and Neural Gas [53]. These algorithms misclassified especially the outliers. Then we tried WKFCM on this dataset. The WKFCM can identify outliers so it gave the best performance as shown in Figure 10.

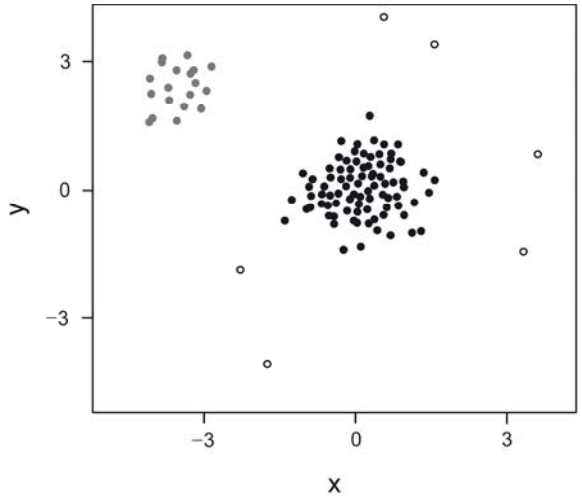


Figure 9 A simulated dataset (Data-2); 86 points in one cluster, 19 points in the other cluster, 6 outliers. Both of the clusters are represented with a different gray level. Filled disks indicate the data points belonging to respective clusters. Circles represent outliers.

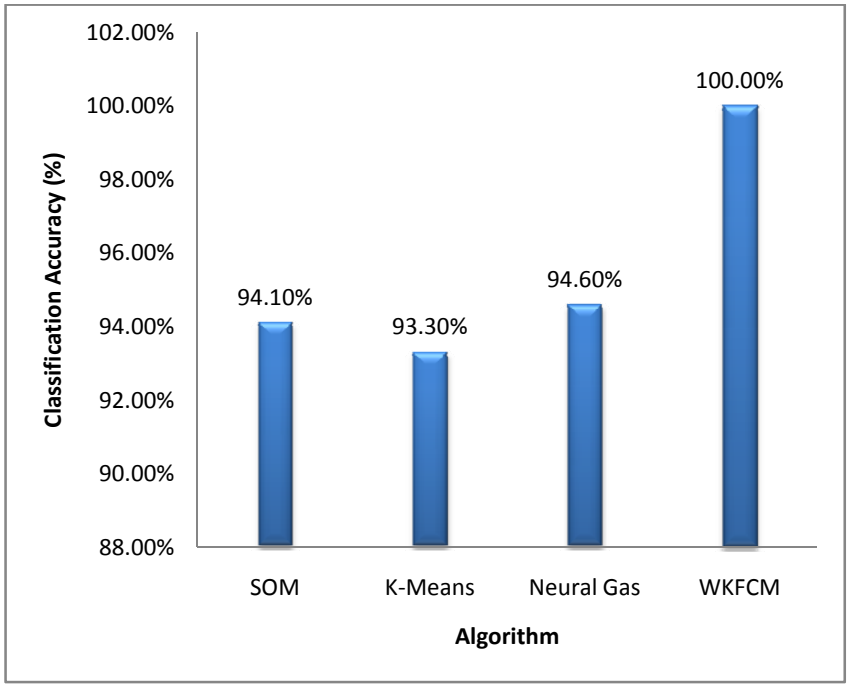


Figure 10 Average WKFCM, SOM, Neural Gas and K-Means performances on simulated Data-2; 111 patterns, 2 features, 2 classes plus outliers. The results have been obtained using ten different runs for each algorithm.

5.2 Experimentation on IRIS Data

IRIS dataset (IRIS dataset can be downloaded from the address: http://www.ics.uci.edu/~mlearn/databases/IRIS/) is the most famous real data benchmark in Machine Learning. IRIS dataset was proposed by Fisher in 1936 [54]. This dataset is formed by 150 points that belong to three different classes. One class is linearly separable from the other two, but the other two are not linearly separable from each other. Since the dimension of IRIS data is 4, IRIS data is usually represented by projecting the data along their principal components. IRIS data projected along the two components is shown in Figures 7(a). We tried WKFCM, *K*-Means, Neural Gas and SOM on IRIS data using three centers, one center for each class. The results using SOM, Neural Gas, *K*-Means and WKFCM are shown in Figure 11.

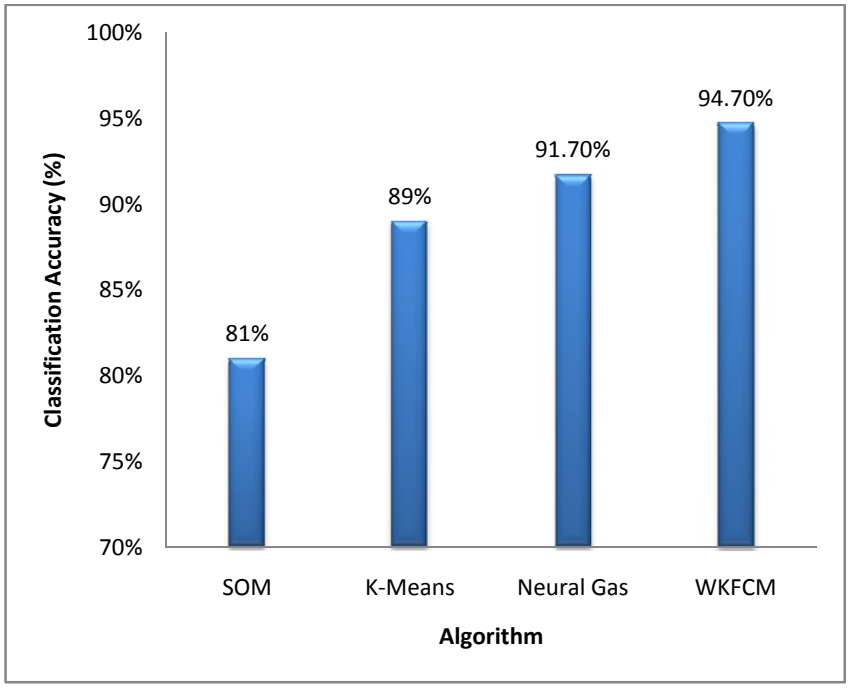


Figure 11 Average WKFCM, SOM, Neural Gas and K-Means performances on IRIS data; 150 patterns, 4 features, 3 classes. The results have been obtained using thirty different runs for each algorithm.

5.3.3 Experimentation on Microarray Data

The clustering performance was firstly evaluated using a Figure Of Merit (FOM), 2-Norm FOM. 2-Norm FOM analysis (as shown in Figures 12, 13 and 14) indicated that no clustering algorithm was the best on all the datasets, with WKFCM, FCM and FSOM being the best, respectively, on the reduced PBM, HR and YCC data.

Whereas, according to the Davies-Bouldin Index analysis (as shown in Figures 15, 16 and 17), WKFCM emerged as the best algorithm on all the three datasets. As WKFCM may generate non-globular clusters with more heterogeneous size distribution, its results for the DBI analysis proved to be the best. Whereas for the FOM analysis, FOM is calculated by averaging the deviations in the left-out condition not cluster by cluster, but by averaging over the whole dataset. Therefore, large clusters with high internal variability have a higher weight in FOM calculation than small, compact clusters.

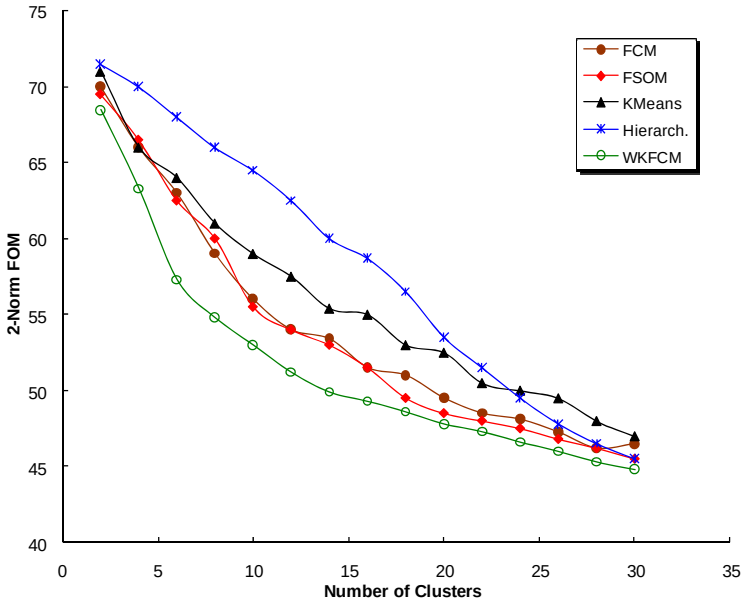


Figure 12 Clustering validation and comparison by 2-Norm FOM–lower values of 2-Norm FOM are better. 2-Norm FOM on the reduced peripheral blood monocyte (PBM) dataset.

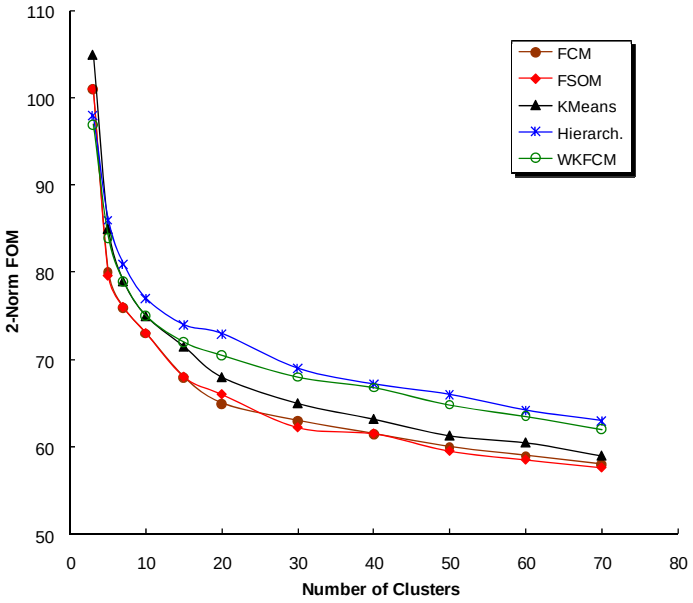


Figure 13 Clustering validation and comparison by 2-Norm FOM–lower values of 2-Norm FOM are better. 2-Norm FOM on the reduced hypoxia response (HR) dataset.

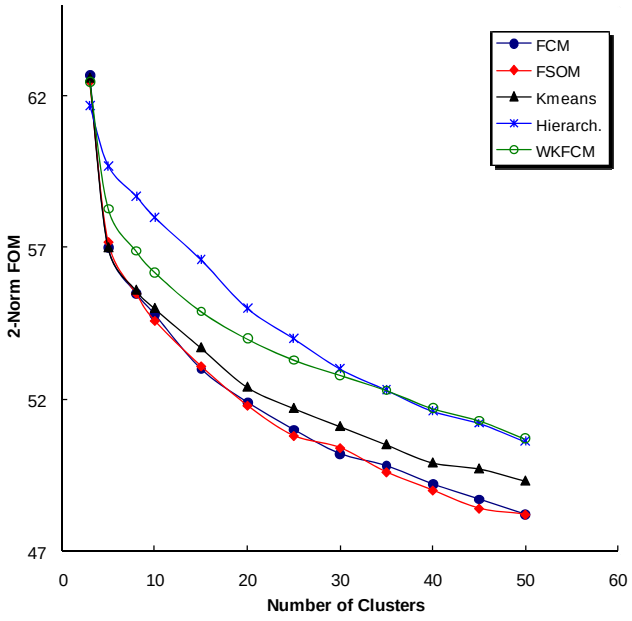


Figure 14 Clustering validation and comparison by 2-Norm FOM–lower values of 2-Norm FOM are better. 2-Norm FOM on the reduced yeast cell cycle (YCC) dataset.

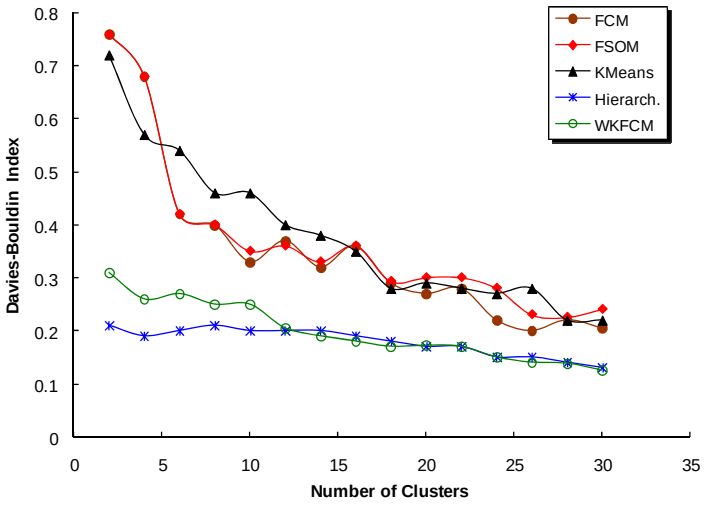


Figure 15 Clustering validation and comparison by Davies-Bouldin Index–lower values of DB Index are better. DB Index on the reduced peripheral blood monocyte (PBM) dataset.

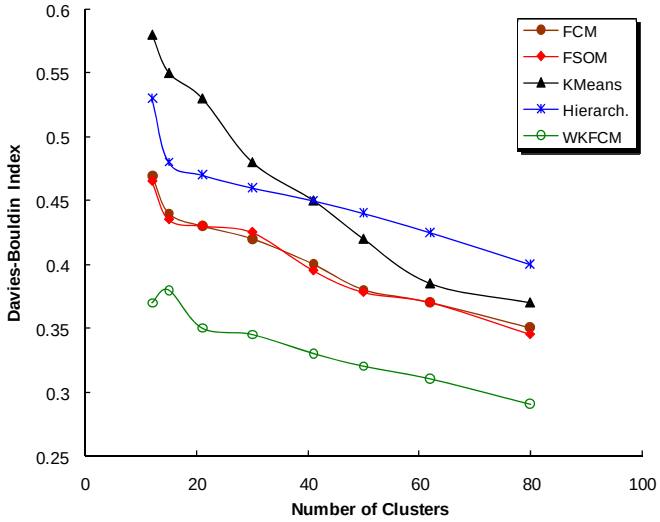


Figure 16 Clustering validation and comparison by Davies-Bouldin Index–lower values of DB Index are better. DB Index on the reduced hypoxia response (HR) dataset.

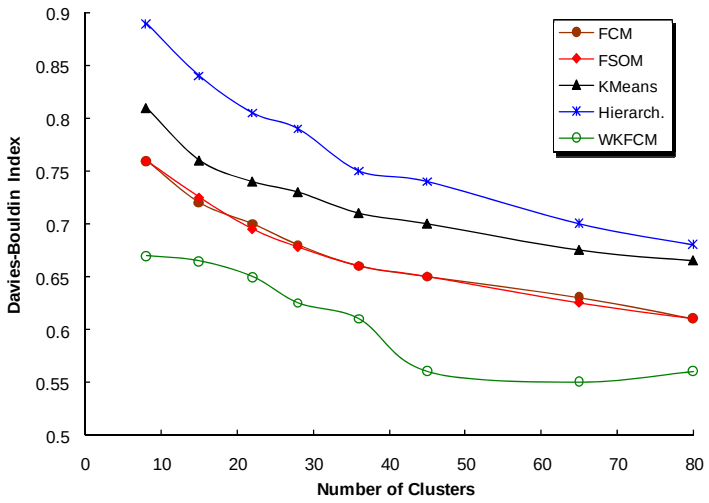


Figure 17 Clustering validation and comparison by Davies-Bouldin Index–lower values of DB Index are better. DB Index on the reduced yeast cell cycle (YCC) dataset.

To provide a quantitative readout of the comparative analysis between the various algorithms, we adopted a procedure to rank the algorithms in the validation analysis based on the area under the index line plots (area under the curve, in a way). The algorithm that had the smallest area under the index line plot was assigned a rank of 1 (the best performance), and the others obtained a progressively higher value of rank (lower performance). The results of this ranking procedure are shown in Table 3. The results illustrate that no single clustering algorithm showed always the best performance on all the datasets and with all validation metrics. However, WKFCM proved to be the best in many cases, and its performance profile across the various datasets and validation metrics, used in this study, is better than those of the other algorithms. This indicates that WKFCM can prove to be an alternative clustering strategy. Furthermore, it can be noted from the results that FCM and FSOM display somewhat similar performance, as these are related algorithms.

Table 3: Ranking of each clustering algorithm across all comparative validation cases (lower value of rank stands for better performance)

Dataset (reduced)	Validation case	WKFCM	Hierarch.	K-Means	FSOM	FCM
PBM	2-Norm FOM	1	5	4	2	3
	DB Index	2	1	5	4	3
HR	2-Norm FOM	4	5	3	2	1
	DB Index	1	4	5	2	3
YCC	2-Norm FOM	4	5	3	1	2
	DB Index	1	5	4	2	3

As in real life, birds of a feather flock together. And, objects do influence their neighboring objects. WKFCM uses this aspect of life. While, in other fuzzy clustering algorithms like Fuzzy C-Means, the fuzzy memberships of data points are determined by their similarity with a series of calculated cluster prototypes. Whereas, WKFCM first uses pairwise similarity measures to define the neighbors of each object and how close each object is to its nearest neighbors, and then it approximates the fuzzy memberships of each object under the influence of its neighbors. In other words, the neighborhood relationships are calculated for all objects, and are used to constrain the fuzzy memberships. In this way, WKFCM performs clustering using not only the expression data, but also the local information extracted from them, which allows reliable capturing of both linear and non-linear relationships. In some sense, this local approximation (by incorporating neighborhood information) acts as a regularizer and biases the solution toward piecewise-homogeneous labeling. Such regularization is also helpful in finding clusters in the data corrupted by noise. Working with these features in the kernel space leads to decent clustering results.

The possible applications of WKFCM can be extended to other than gene expression datasets. WKFCM can be applied to any dataset if a neighborhood can be defined for each object. In comparison with the other clustering algorithms, WKFCM is more robust with respect to outliers and noise, since it has a mechanism that permits discarding outliers and noise. However, one main quality of WKFCM lies in producing nonlinear separation surfaces among data. WKFCM can separate classes of data that are not linearly separable by the other clustering algorithms.

WKFCM's main limitation is the computation time required by the algorithm. However, the availability of faster machines and low cost of memory encourages the applicability of WKFCM in real world applications.

6. Conclusions

A Kernel based Method, Weighted Kernel Fuzzy C-Means incorporating local approximation (WKFCM), has been presented in this paper. WKFCM is especially suitable for clustering data with fuzzy structures, having nonlinearly separable clusters, such as microarray gene expression data.

WKFCM is a new algorithm that we specifically tested on microarray gene expression data. It brings significant improvements in the partitioning of genes based on their expression profiles. Its good performance is derived from a combination of advantageous features, some of which are distinctive, like the ability to capture dataset-specific structures by using kernel transformation, and by defining neighborhood relations and the subsequent neighborhood approximation of fuzzy memberships, so that non-globular and non-linear clusters can also be captured. Particularly, the neighborhood approximation, along with distinctive features of kernel methods, makes WKFCM distinct from all other clustering approaches. It has the mechanism for identifying outlier genes whose expression patterns do not allow reliable assignment to any cluster. Our results also confirm that no clustering strategy is always the best for any data type, which keeps the avenues of choice among different algorithms open. These results encourage the use of WKFCM for the

solution of real world problems. Future work includes extension of experimental validation to image segmentation.

Acknowledgements

This work was supported in part by the Ministry of Higher Education under Fundamental Research Grant Scheme (Project Number 78096).

CHAPTER 3

CONCLUSIONS

3.1 Introduction

Traditionally, theory and algorithms of machine learning and statistics has been very well developed for the linear case. Linear modeling techniques explicitly assume linear relations between the input and output variables, but in many real-life case studies, the relations are typically observed to be nonlinear. In kernel methods, the implicit kernel induced feature space interpretation allows to extend the linear methods to kernel methods for nonlinear modeling. In this study we have investigated Kernel Methods for *Clustering*, namely Kernel Methods that do not require target data.

3.2 Conclusion

In this study, a Weighted Kernel Fuzzy C-Means (WKFCM) has been presented. WKFCM is especially suitable for clustering data with fuzzy structures, such as microarray gene expression data.

WKFCM is a new algorithm, that we specifically tested on microarray gene expression data, that brings significant improvements in the partitioning of genes based on their expression profiles. Its good performances are derived from a

combination of advantageous features, some of which are distinctive, like the ability to capture dataset-specific structures by defining neighborhood relations and the subsequent approximation of fuzzy memberships influenced by neighborhood, so that non-globular and non-linear clusters can also be captured and do not get fragmented by the process. In particular, it is the novelty of neighborhood approximation that makes WKFCM distinct from all other clustering approaches. It has the mechanism for defining outlier genes whose expression patterns do not allow reliable assignment to any cluster. Other interesting features are common to fuzzy clustering algorithms, like non-univocal assignment of memberships to genes. Our results also confirm that no clustering strategy is always the best for any data type, which renders the choice between different algorithms.

3.3 Future Work

As WKFCM is not computationally very efficient, therefore a first line of research involves the optimization or efficient implementation of the WKFCM. Another future research line is the development of the specific application oriented kernels, instead of the gaussian one, that can be used in the WKFCM. Another future work could be the extension of WKFCM for clustering incomplete data.

Finally the application of the WKM for the solution of real problems will be performed in the next future. And, it will prove to be in line with the national thrust of prosperous Malaysia.

REFERENCES

1. Hofmann, T., B. Schölkopf, and A.J. Smola, *Kernel methods in machine learning.* To appear in: Annals of Statistics 2007. **156**(2007).
2. Vapnik, V., *Statistical Learning Theory*. 1998: John Wiley & Sons.
3. Cristianini, N. and J.S. Taylor, *An Introduction to Support Vector Machines*. 2000: Cambridge Academic Press.
4. Schölkopf, B., A.J. Smola, and K.R. Müller, *Nonlinear component analysis as a kernel eigenvalue problem.* Neural Computation, 1998. **10**(5): p. 1299–1319.
5. Cover, T.M., *Geomeasureal and statistical properties of systems of linear inequalities in pattern recognition.* IEEE Transactions on Electronic Computers, 1965. **EC-14**(3): p. 326–334.
6. Dhillon, I.S., Y. Guan, and B. Kulis, *Weighted Graph Cuts without Eigenvectors: A Multilevel Approach.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2007. **29**(11): p. 1944–1957.
7. Girolami, M., *Mercer kernel based clustering in feature space.* IEEE Transactions on Neural Networks, 2002. **13**(3): p. 780–784.
8. Jain, A.K., M.N. Murty, and P.J. Flynn, *Data clustering: a review.* ACM Computing Surveys, 1999. **31**(3): p. 264–323.
9. Camastra, F. and A. Verri, *A novel kernel method for clustering.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2005. **27**(5): p. 801–805.
10. Filippone, M., et al., *A survey of kernel and spectral methods for clustering.* Pattern Recognition, 2008. **41**: p. 176–190.

11. Handl, J., J. Knowles, and D.B. Kell, *Computational cluster validation in post-genomic data.* Bioinformatics, 2005. **21**: p. 3201–3212.

12. Hsu, C.-W., C.-C. Chang, and C.-J. Lin, *A practical guide to support vector classification.*, in *Technical Report*. 2003, Department of Computer Science and Information Engineering, National Taiwan University.

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

14. Allison, D.B., et al., *Microarray data analysis: from disarray to consolidation and consensus.* Nature Reviews Genetics, 2006. **7**(1): p. 55–65.

15. Chi, J.T., et al., *Gene Expression Programs in Response to Hypoxia: Cell Type Specificity and Prognostic Significance in Human Cancers.* PLoS Medicine, 2006. **3**(3): p. e47.

16. D'haeseleer, P., *How does gene expression clustering work?* **.** Nature Biotechnology, 2005. **23**: p. 1499–1501.

17. Eisen, M.B., et al. *Cluster analysis and display of genome-wide expression patterns.* in *Proceedings of the National Academy of Sciences*. 1998. USA.

18. Golub, T.R., et al., *Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring.* Science, 1999. **286**: p. 531–537.

19. Xing, E.P. and R.M. Karp, *CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts.* Bioinformatics, 2001. **17**: p. S306–S315.

20. Yeung, K., et al., *Model-based clustering and data transformations for gene expression data.*, in *Technical Report UW-CSE-01-04-02*. 2002, Department of Computer Science and Engineering, University of Washington.

21. Herrero, J., A. Valencia, and J. Dopazo, *A hierarchical unsupervised growing neural network for clustering gene expression patterns.* Bioinformatics, 2001. **17**: p. 126–136.

22. Lukashin, A.V. and R. Fuchs, *Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters.* Bioinformatics, 2001. **17**: p. 405–114.

23. Hastie, T., et al., *'Gene shaving' as a method for identifying distinct sets of genes with similar expression patterns.* Genome Biology, 2000. **1**: p. 1–21.

24. Kluger, Y., et al., *Spectral biclustering of microarray data: coclustering genes and conditions.* Genome Research, 2003. **13**(4): p. 703–716.

25. Madeira, S.C. and A.L. Oliveira, *Biclustering algorithms for biological data analysis: a survey.* IEEE Transactions on Computational Biology and Bioinformatics, 2004. **1**(1): p. 24–45.

26. Di Gesu, V., et al., *GenClust: A genetic algorithm for clustering gene expression data.* BMC Bioinformatics, 2005. **6**: p. 289.

27. Garge, N.R., et al., *Reproducible clusters from microarray research: whither?* BMC Bioinformatics, 2005. **6**(Suppl 2): p. S1.

28. Kohonen, T., *Self-Organizing Maps*. 1997, New York, USA: Springer.

29. Chen, Y.D., M.L. Bittner, and E.R. Dougherty, *Issues associated with microarray data analysis and integration.* Nature Genetics, 1999. **22**: p. 213–215.

30. Costa, I.G., F.A. de Carvalho, and M.C. de Souto, *Comparative analysis of clustering methods for gene expression time course data.* Genetics and Molecular Biology, 2004. **27**: p. 623–631.

31. Datta, S. and S. Datta, *Comparisons and validation of statistical clustering techniques for microarray gene expression data.* Bioinformatics, 2003. **19**: p. 459–466.

32. Gelmi, C.A., *A novel probabilistic framework for microarray data analysis: from fundamental probability models to experimental validation. PhD thesis*. 2006, University of Delaware.

33. Bezdek, J.C., et al., *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. 1999, Boston: Kluwer Academy Publishers.

34. Dembele, D. and P. Kastner, *Fuzzy c-means method for clustering microarray data.* Bioinformatics, 2003. **19**: p. 973–980.

35. Gasch, A.P. and M.B. Eisen, *Exploring the conditional coregulation of yeast gene expression through fuzzy k-means clustering.* Genome Biology, 2002. **3**(11): p. 1–22.

36. Pascual-Marqui, R.D., et al., *Smoothly distributed fuzzy c-means: a new self-organizing map.* Pattern Recognition, 2001. **34**: p. 2395–2402.

37. Belacel, N., et al., *Fuzzy J-Means and VNS methods for clustering genes from microarray data.* Bioinformatics, 2004. **20**: p. 1690–1701.

38. Qu, Y. and S. Xu, *Supervised cluster analysis for microarray data based on multivariate Gaussian mixture.* Bioinformatics, 2004. **20**: p. 1905–1913.

39. Guan, Y., *Large-Scale Clustering: Algorithms and Applications*. 2006, PhD Dissertation: The University of Texas at Austin.

40. Ben-Hur, A., et al., *Support vector clustering.* Journal of Machine Learning Research, 2001. **2**: p. 125–137.

41. Roweis, S.T. and L.K. Saul, *Nonlinear dimensionality reduction by locally linear embedding.* Science, 2000. **290**: p. 2323–2326.

42. Saul, L.K. and S.T. Roweis, *Think globally, fit locally: unsupervised learning of low dimensional manifolds.* Journal of Machine Learning Research, 2003. **4**: p. 119–155.

43. Futschik, M.E., *Methods for knowledge discovery in microarray data. PhD thesis*. 2003, University of Otago: Dunedin, New Zealand.

44. Mehta, T., M. Tanik, and D.B. Allison, *Towards sound epistemological foundations of statistical methods for high-dimensional biology.* Nature Genetics, 2004. **36**(9): p. 943–947.

45. Troyanskaya, O., et al., *Missing value estimation methods for DNA microarrays.* Bioinformatics, 2001. **17**(6): p. 520–525.

46. Quackenbush, J., *Microarray data normalization and transformation.* Nature Genetics, 2002. **32**: p. 496–501.

47. Schuchhardt, J., et al., *Normalization strategies for cDNA microarrays.* Nucleic Acids Research, 2000. **28**(10): p. e47.

48. Yeung, K.Y., D.R. Haynor, and W.L. Ruzzo, *Validating clustering for gene expression data.* Bioinformatics, 2001. **17**(4): p. 309–318.

49. Yeung, K.Y., D.R. Haynor, and W.L. Ruzzo, *Validating clustering for gene expression data.*, in *Technical Report UW-CSE-00-01-01*. 2000, Department of Computer Science and Engineering, University of Washington.

50. Davies, D.L. and D.W. Bouldin, *A cluster separation measure.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979. **1**(2): p. 224–227.

51. Spellman, P.T., et al., *Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization.* Molecular Biology of the Cell, 1998. **9**(12): p. 3273–3297.

52. Hartuv, E., et al., *An algorithm for clustering cDNA fingerprints.* Genomics, 2000. **66**: p. 249–256.

53. Martinetz, T.E. and K.J. Schulten, *Neural-gas network for vector quantization and its application to time-series prediction.* IEEE Transactions on Neural Networks, 1993. **4**(4): p. 558-569.

54. Fisher, R.A., *The use of multiple measurements in taxonomic problems.* Annals of Eugenics, 1936. **7**: p. 179–188.

APPENDIX A

LIST OF PUBLICATIONS

1. M.N. Md. Sap and A.M. Awan, "A weighted fuzzy kernel based method incorporating local approximation for clustering microarray data," Submitted to *Journal of Biomedical Informatics*.

2. M.N. Md. Sap and A.M. Awan, "A new weighted kernel clustering method incorporating local approximation," Submitted to *Pattern Recognition*.

3. M.N. Md. Sap and A.M. Awan, "A New Weighted Kernel-Based Clustering Method Incorporating Local Approximation", *Journal of Information Technology*, UTM, June 2007.

4. M.N. Md. Sap and Shafaatunnur, "Clustering Spatial Data based on Artificial Neural Networks: A Review", *Journal of Information Technology*, UTM, Dec 2007.

5. M.N. Md. Sap and Mojtaba Kohram, "Support Vector Machines: Trends and Applications", in *Proc. Postgraduate Annual Research Seminar 2007* (PARS'07), FSKSM, University Technology Malaysia, June 2007.