Chapter 1

Introduction

There are certain basic attributes which define a human being. Focusing on the world's human population, two traits which are shared among humans might be that all humans are living organisms and live on planet earth. The human population can easily be divided into two separate sub-populations (groups) based on the gender attribute. Both of the male and female populations can be further divided on the age attribute into multiple sub-populations. The end result might be a group of humans which share common traits such as teenage males between the age of 12 and 20. Another group might represent female humans above the age of 50. Therefore, humans forming part of the same group share some common trait compared to humans which form part of other groups. Thus, natural clustering exists in our daily lives. Whether the attribute is based on choice of music genre, political interest and/or religion, there is a spontaneous and natural clustering in society which is determined by the similarity or dissimilarity of different attributes. Since teenage males will age with time they will eventually form part of a different group. This implies that the population is non-stationary. On a smaller scale, spontaneous clustering also exists in the natural immune system where lymphocytes co-operate by co-stimulating each other in response to an invading antigen. The end result is the formation of lymphocyte networks (groups) with a similar structure to react to the invading antigen. Since the body is also frequently exposed to unseen and unfamiliar antigens, the natural immune system needs to adapt to changes in the antigen structure. Thus, the antigen population is also non-stationary. This thesis proposes an artificial immune model which is based on the network theory of co-stimulating lymphocytes and is applied to the problem of data clustering in stationary and non-stationary environments.

1.1 Motivation

Clustering of observations or features into different partitions in order to discover hidden traits in the data is of considerable value. The discovered traits could influence the strategic decisions of a business, the effect of medicine on certain diseases or highlight emigration/immigration patterns of citizens in a country. It is clear that clustering is a fundamental cornerstone in decision making within various disciplines. Many of the existing network theory based artificial immune systems have been applied to data clustering. The formation of artificial lymphocyte (ALC) networks represents potential clusters in the data. Although these models do not require any user specified parameter of the number of required clusters to cluster the data, these models do have a drawback in the techniques used to determine the number of ALC networks. Another drawback is that these models have a large number of user parameters which control the outcome of the clustering performance. Specifying the optimal set of values for these control parameters is a time-consuming and challenging task, since there could be an optimal set for each data set that needs to be clustered and the optimal set of values has a high probability to change in a non-stationary environment. Furthermore, the techniques utilised by these models to determine the number of ALC networks are either based on a network affinity threshold with a proximity matrix of network affinities between the ALCs in the population or a hybrid approach is taken by clustering the ALC population using a clustering algorithm. Specifying the correct network affinity threshold to obtain the correct or required number of clusters can be a formidable task, especially in a non-stationary environment. A potential drawback to a hybrid approach is that the formed sub-nets might not always contain ALCs with a good or generic representation of the data. Furthermore, both of these techniques are computationally expensive. This thesis proposes a network based artificial immune model which is applied to data clustering in stationary and non-stationary environments. The proposed model is independent of a network affinity threshold and do not need to follow a hybrid approach to determine the number of clusters. Furthermore, the proposed model has considerable less control parameters in comparison to existing network based AIS models. Also, the proposed model is enhanced to dynamically determine the number of clusters in a data set.

1.2 Objectives

The primary objectives of this thesis can be summarised as follows:

- To develop an alternative artificial lymphocyte network topology which is independent of

a network affinity threshold and do not need to follow a hybrid approach to determine the number of clusters.

- To develop a theoretical network based artificial immune model which utilises the alternative network topology for data clustering.
- To develop two techniques which can be used with the proposed artificial immune model to dynamically determine the number of clusters in a data set.
- To develop a method for the generation of synthetic non-stationary data which is based on different data migration types.
- To show that the proposed model can be applied to data clustering of non-stationary environments.

1.3 Methodology

The algorithms developed in this thesis are first presented and discussed. Empirical results were obtained using a selection of data clustering problems with known characteristics and which covers a good distribution of problems in stationary environments. The results of two classical clustering algorithms and three network based artificial immune models were also reported for the same selection of stationary data clustering problems. These results showed the relative clustering performance of the proposed model when compared to other existing clustering and network based artificial immune models. The same selection of stationary data clustering problems was used for the purpose of evaluating the capability of the enhanced version of the proposed model to dynamically determine the number of clusters in a data set. Results of the enhanced models were also compared to the results obtained from a classical clustering model to show the relative clustering performance of the enhanced models. Furthermore, the time complexity of these models was also discussed. Various synthetic non-stationary data clustering problems with known characteristics were also used to evaluate the clustering performance of the proposed model in a non-stationary environment. The results of two network based artificial immune models were also reported for the same synthetic non-stationary data clustering problems. These results showed the relative clustering performance of the proposed model when compared to other existing network based artificial immune models. All the reported results are averages and standard deviations taken over 50 runs, since the proposed model is population based and

has a stochastic nature. All parameter values for the respective algorithms were found empirically to deliver the best performance for clustering the applicable data set. A non-parametric Mann-Whitney U hypothesis test between the clustering quality of the proposed model and the clustering quality of each of the other models was used to investigate whether there is a statistical significant difference between the clustering quality of two models for a specific data set or not.

1.4 Contributions

The main contributions of this thesis are:

- The development of a novel network based artificial immune model which utilises an index based artificial neighbourhood network topology for data clustering of stationary environments. The developed model has less control parameters than existing network based artificial immune models, is independent of a network affinity threshold and does not need to follow a hybrid approach to determine the number of clusters.
- The development of two techniques which enhances the proposed network based artificial immune model to dynamically determine the number of clusters in a data set.
- The development of a simple method to generate synthetic non-stationary data which follows different data migration types.
- The application of the proposed network based artificial immune model to the clustering of non-stationary environments.
- Empirical analysis of the behaviour of all versions of the proposed network based artificial immune model under different parameter settings.

The following list of published or currently reviewed articles support the main contributions of this thesis:

A.J. Graaff and A.P. Engelbrecht. Chapter 18: Natural Immune System. *Computational Intelligence: An Introduction*, 2nd Edition, A.P. Engelbrecht (Author), John Wiley & Sons, October 2007.

A.J. Graaff and A.P. Engelbrecht. Chapter 19: Artificial Immune Models. *Computational Intelligence: An Introduction*, 2nd Edition, A.P. Engelbrecht (Author), John Wiley & Sons, October 2007.

A.J. Graaff and A.P. Engelbrecht. A local network neighbourhood artificial immune system for data clustering. In *IEEE Congress on Evolutionary Computation*, CEC 2007., pp. 260–267, 2007.

A.J. Graaff and A.P. Engelbrecht. Towards a self regulating local network neighbourhood artificial immune system for data clustering. In *IEEE Congress on Evolutionary Computation*, CEC 2008.(IEEE World Congress on Computational Intelligence), pp. 633–640, 2008.

A.J. Graaff and A.P. Engelbrecht. Optimised Coverage of Non-self with Evolved Lymphocytes in an Artificial Immune System. *International Journal of Computational Intelligence Research*, vol. 2, no. 2, pp. 127–150, 2006.

A.J. Graaff and A.P. Engelbrecht. Clustering Data in an Uncertain Environment using an Artificial Immune System. *Pattern Recognition Letters*, vol. 32, no. 2, pp. 342–351, January 2011.

A.J. Graaff and A.P. Engelbrecht. Using sequential deviation to dynamically determine the number of clusters found by a local network neighbourhood artificial immune system. *Applied Soft Computing*, vol. 11, pp. 2698–2713, March 2011.

A.J. Graaff and A.P. Engelbrecht. Clustering Data in Stationary Environments with a Local Network Neighborhood Artificial Immune System. *International Journal of Machine Learning and Cybernetics*, submitted May 2011.

1.5 Thesis Outline

The thesis is organised as follows:

- *Chapter 2* discusses the problem of data clustering. A formal definition of data clustering is given with an elaboration on different similarity measures and existing clustering approaches. This is followed by an overview of different clustering performance measures to evaluate the partitioning quality of a clustering algorithm applied to stationary data. The different performance measures applied to optimisation algorithms for problems in non-stationary environments are then discussed. These performance measures are used to quantify and define performance measures that can be used to evaluate the partitioning quality of clustering algorithms in non-stationary environments. Furthermore, a brief introduction to outliers and outlier detection is given as well as a discussion of two alternative computational models which can be applied to the problem of data clustering.

- *Chapter 3* reviews the functional process of the natural immune system. The different theories in immunology regarding the functioning and organisational behavior between lymphocytes are discussed. These theories include the classical view, clonal selection theory, network theory, and danger theory. The classical view is first discussed in detail, since the other theories are based on concepts and elements within the classical view. The classical view forms a base onto which the other theories are explained. A brief review of the dendritic cell system is also given.

- *Chapter 4* discusses some of the most familiar artificial immune system (AIS) models which are inspired by the different theories in the science of immunology. The chapter highlights the basic components of an AIS model and introduces the shape space model. Furthermore, an overview of different measures of affinity between an artificial lymphocyte and an antigen pattern within a specific shape space is given which is followed by an overview on the different matching rules to determine whether an ALC binds to an antigen pattern. The remainder of the chapter briefly discusses some of the AIS models which are respectively inspired by the negative selection, clonal selection and danger theories. Since the proposed AIS model in this thesis is inspired by and mostly based on the network theory, a more detailed overview is given on existing network based AIS models within the context of data clustering. Also, different theoretical approaches to determine the possible interactions in an ALC network are discussed.

- *Chapter 5* presents a novel network theory inspired artificial immune system. Specifically, the network topology of co-stimulated lymphocytes inspired the modelling of the local network neighbourhood artificial immune system (LNNAIS). The chapter introduces the concept of an index based neighbourhood topology which is utilised by LNNAIS to determine the network connectivity between ALCs. Each of the formed local ALC neighbourhood structures represents a cluster in a data set. The differences and similarities between existing network based AIS models and the proposed LNNAIS model are also discussed. The proposed LNNAIS model is compared to classical clustering algorithms and existing network based AIS models which are applied to data clustering problems. Furthermore, a sensitivity analysis is also done on the proposed model to investigate the influence of the model's parameters on the quality of the clusters.

- *Chapter 6* proposes two techniques which can be used with the proposed local network neighbourhood artificial immune model to dynamically determine the number of clusters in a data set. The first technique utilises cluster validity indices and is similar to the multiple

execution approach, though computationally less expensive. The second technique is based on sequential deviation outlier detection. The results of a multiple execution approach of K-means clustering is compared to the results obtained from both the proposed LNNAIS techniques to dynamically determine the number of clusters in a data set. The influence of the parameters of LNNAIS on the number of dynamically determined clusters in a data set is also investigated.

- *Chapter 7* defines and discusses different non-stationary environments. A technique to generate synthetic data sets for each of the defined non-stationary environments is proposed. Different synthetic data sets are then generated based on the defined non-stationary environments. The proposed local network neighbourhood artificial immune model and the enhanced version of the model to dynamically determine the number of clusters are applied to the clustering of the generated synthetic non-stationary data. The results are compared to the results obtained from two existing network based artificial immune models to cluster the non-stationary data. The influence of the different non-stationary environments on the parameters of the proposed model is also investigated.

- *Chapter 8* highlights the conclusions of this thesis and presents ideas relating to possible future work.

- *Appendix A* lists and defines the symbols used throughout this thesis.

- *Appendix B* lists the publications derived from this thesis.

Chapter 2

Clustering and Quality Measures

This chapter gives a formal definition of data clustering. A brief overview of different clustering techniques is given. Different similarity measures are discussed as well as different cluster quality measures. These quality measures are then discussed in the context of dynamic and uncertain environments, i.e. clustering non-stationary data.
The chapter is organised as follows:

- Section 2.1 gives a formal definition of data clustering and the notation used by the rest of the chapter.
- Section 2.2 discusses the different distance-based *similarity* measures.
- Section 2.3 discusses the most familiar clustering algorithms which are categorised into *hierarchical* clustering or *partitional* clustering methods.
- Section 2.4 introduces the different categories of cluster validity measures to evaluate the partitioning quality of a clustering algorithm. The section discusses the cluster validity indices which form part of the *relative* criteria.
- Section 2.5 introduces different measures to evaluate the performance of an optimisation algorithm which is applied to problems in non-stationary environments. These performance measures are used to quantify and define measures that can be used to evaluate the partitioning quality of clustering algorithms in non-stationary environments.
- Section 2.6 defines outliers and explains three different approaches for outlier detection.
- Section 2.7 discusses two alternative computational algorithms which can be applied to the problem of data clustering.

- Section 2.8 concludes the chapter by giving an overall summary of the chapter and discussing the relevance of each section to the work in this thesis.

2.1 Data Clustering

Patterns in a data set can be structured into different groups in such a way that patterns within the same group are more *similar* compared to patterns across different groups. Each of the formed clusters is represented by a *centroid* [122]. Data clustering can formally be defined as follows [15, 96]:
Let P be the data set of patterns in N-dimensional space that needs to be clustered. Thus, $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_{I-1}, \mathbf{p}_I\}$ where $\mathbf{p}_i$ is an N-dimensional *feature vector* (pattern) and I is the number of feature vectors. The partitioning of P into K clusters, $\{C_1, C_2, \ldots, C_K\}$, satisfies the following conditions:

1. $|C_k| \neq 0, k = 1, 2, \ldots, K$, meaning that clusters are not allowed to be empty;
2. $P = \cup_{k=1}^{K} C_k$, meaning that each feature vector is assigned to a cluster;
3. $|C_k \cap C_j| = 0, k \neq j$, meaning that each feature vector is assigned to only one cluster (in the case of *crisp* or *hard* clustering, i.e. *exclusive clustering*); or
4. $|C_k \cap C_j| > 0, k \neq j$, meaning that each feature vector can be assigned to more than one cluster with a certain degree. *Fuzzy* clustering is an example of *overlapping clustering* for which this condition holds.

The most general measure of *similarity* or *dissimilarity* between feature vectors is based on the distance between these vectors (e.g. Euclidean distance). A cluster's centroid can describe a specific *concept*. Feature vectors with a *similar* or common concept are grouped together. Clustering algorithms are applied to data clustering and compression [26, 64, 174], image segmentation [95, 145, 151], and vector and color image quantization [9, 145, 185]. The following section discusses some of the distance-based similarity measures between feature vectors.

2.2 Similarity Measures

This section discusses the different distance-based *similarity* measures. One of these distance measures is the Minkowski distance between multidimensional feature vectors, defined as [96]

$$\sigma_\varepsilon\left(\mathbf{p}_i,\mathbf{p}_j\right) = \left[\sum_{n=1}^{N}\left(\mathbf{p}_{i,n}-\mathbf{p}_{j,n}\right)^\varepsilon\right]^{\frac{1}{\varepsilon}} \tag{2.1}$$

$$= \left\|\mathbf{p}_i-\mathbf{p}_j\right\|^\varepsilon \tag{2.2}$$

where N is the dimensions of feature vectors $\mathbf{p}_i$ and $\mathbf{p}_j$. The Euclidean distance is derived from the Minkowski measure by setting $\varepsilon = 2$ [15, 96]. The Euclidean distance is the most commonly used *similarity* measure, which is defined as

$$\sigma_2\left(\mathbf{p}_i,\mathbf{p}_j\right) = \left[\sum_{n=1}^{N}\left(\mathbf{p}_{i,n}-\mathbf{p}_{j,n}\right)^2\right]^{\frac{1}{2}} \tag{2.3}$$

$$= \sqrt{\sum_{n=1}^{N}\left(\mathbf{p}_{i,n}-\mathbf{p}_{j,n}\right)^2} \tag{2.4}$$

$$= \left\|\mathbf{p}_i-\mathbf{p}_j\right\|^2 \tag{2.5}$$

The Manhattan distance between two feature vectors is the sum of the absolute differences of their features (attributes) and can be derived from the Minkowski measure by setting $\varepsilon = 1$ [15]. The Manhattan distance is defined as

$$\sigma\left(\mathbf{p}_i,\mathbf{p}_j\right) = \sum_{n=1}^{N}\left|\mathbf{p}_{i,n}-\mathbf{p}_{j,n}\right| \tag{2.6}$$

The distance between two feature vectors can also be calculated as the maximum absolute difference between the values of each dimension. This distance measure is known as the Chebychev distance, defined as [132]

$$\sigma\left(\mathbf{p}_i,\mathbf{p}_j\right) = \max_{n=1,\ldots,N}\left|\mathbf{p}_{i,n}-\mathbf{p}_{j,n}\right| \tag{2.7}$$

The Chebychev distance is more appropriate in cases where the (dis)similarity between two feature vectors is reflected in individual dimensions. The Chebychev distance is also sensitive to outliers.

A drawback of the Minkowski distance measure (including all the derivatives like Euclidean

distance) is what is known as the 'curse of dimensionality' [14]. The general understanding of the 'curse of dimensionality' is that with an increase in dimensionality of space the distribution of distances between the feature vectors in space becomes uniform [1]. In the context of data clustering this means that the larger the dimensionality of the search space, the larger the total space that needs to be explored in order to capture a part of the data. The Manhatten distance measure is however more preferable than the Euclidean distance measure for high dimensional data [1].

Another measure of similarity between two feature vectors is the cosine similarity measure [15]. Different to the previously discussed distance measures, the cosine similarity measures the angle between two feature vectors. The cosine similarity, Γ, between two feature vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ is defined as

$$\Gamma\left(\mathbf{p}_i, \mathbf{p}_j\right) = arccos\left(\frac{\mathbf{p}_i \bullet \mathbf{p}_j}{\|\mathbf{p}_i\| \|\mathbf{p}_j\|}\right) \tag{2.8}$$

where $\mathbf{p}_i \bullet \mathbf{p}_j$ is the *dot product* between vectors $\mathbf{p}_i$ and $\mathbf{p}_j$ and $\Gamma \in [0, \pi]$. The value of Γ indicates the degree of similarity or dissimilarity between two vectors. Γ values closer to 0 imply a higher similarity between two vectors, and Γ values closer to π imply a higher dissimilarity between two vectors. Thus, if $\Gamma = \pi$, then the two vectors are exact opposites from one another. $\Gamma = \frac{\pi}{2}$ means that the two vectors are independent, and when $\Gamma = 0$ the two vectors are exactly the same. An advantage of the cosine similarity measure compared to the Minkowski measure is that the dissimilarity (distance) does not increase with an increase in the number of dimensions. The cosine similarity measure is therefore not influenced by the 'curse of dimensionality', making the cosine similarity measure more appropriate for clustering data of high dimensionality.

The Mahalanobis distance calculates the probability that a feature vector belongs to a set of given feature vectors [15, 96]. The distance between the feature vector and the average of the set gives an indication of the probability that a feature vector belongs to the set. Thus, a closer distance to the average has a higher probability of membership. The Mahalanobis distance can also measure the dissimilarity between two feature vectors, and is defined as

$$\sigma\left(\mathbf{p}_i, \mathbf{p}_j\right) = \sqrt{\left(\mathbf{p}_i - \mathbf{p}_j\right)^T \mathbf{Z}^{-1} \left(\mathbf{p}_i - \mathbf{p}_j\right)} \tag{2.9}$$

where $\mathbf{Z}$ is the covariance matrix of the given set of feature vectors and $\left(\mathbf{p}_i - \mathbf{p}_j\right)^T$ is the transpose of vector $\left(\mathbf{p}_i - \mathbf{p}_j\right)$. Thus, from the above definition, each feature vector is given a weight

which is based on the vector's variance. The Mahalanobis distance is only appropriate to a set of feature vectors with a multivariate Gaussian distribution.

For all of the previously discussed distance measures, it is assumed that a feature vector consists of continuous features (attributes), i.e. $\mathbf{p}_{i,n} \in \Re, \forall n$ where n is the n-th attribute of feature vector $\mathbf{p}_i$. In cases where attributes are nominal-valued, the Hamming distance is used to measure similarity or rather dissimilarity [73]. The Hamming distance between two feature vectors of equal length is the number of positions which are different between the two vectors, defined as

$$\sigma\left(\mathbf{p}_i, \mathbf{p}_j\right) = \sum_{n=1}^{N} 1 \quad \forall \mathbf{p}_{i,n} \neq \mathbf{p}_{j,n} \tag{2.10}$$

Thus for feature vectors in binary space, i.e. $\mathbf{p}_i \in \{0,1\}^N, \forall i$, the above function can be re-defined as

$$\sigma\left(\mathbf{p}_i, \mathbf{p}_j\right) = \sum_{n=1}^{N} \oplus\left(\mathbf{p}_{i,n}, \mathbf{p}_{j,n}\right) \tag{2.11}$$

where $\oplus$ is the exclusive-or between the bits of $\mathbf{p}_i$ and $\mathbf{p}_j$, n is the bit-index and N is the size of the binary string (dimensions).

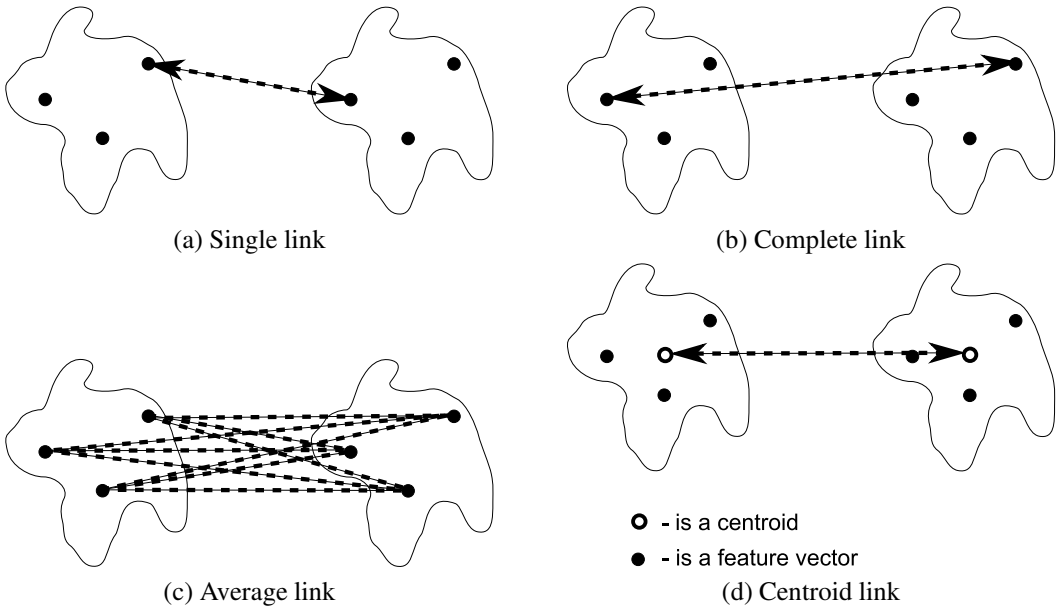
Another familiar similarity measure not covered in this section is *Pearson's* correlation coefficient. The interested reader is referred to [75, 83] for more information.

2.3 Clustering Algorithms

Data clustering algorithms can be categorised into *hierarchical* clustering or *partitional* clustering methods. This section highlights the differences between the aforementioned categories and discusses the most familiar clustering algorithms found in each category.

2.3.1 Hierarchical Clustering

Hierarchical clustering methods iteratively partition a data set into a hierarchy of clusters. This means that each level of the hierarchy consists of a number of clusters, which are obtained by further partitioning of the clusters in the preceding level. A hierarchical clustering algorithm is either *agglomerative* or *divisive* [56, 96]. In both cases, a similarity measurement is used to either merge clusters or divide clusters, generating a tree-like structure.



(a) Single link (b) Complete link

(c) Average link (d) Centroid link

Figure 2.1 Linking Techniques in Hierarchical Clustering

In *agglomerative* hierarchical clustering, each feature vector in the data set initially represents a cluster [96]. In each iteration, *similar* clusters are merged. The process continues until only one cluster is left [96]. Thus, *agglomerative* algorithms follow a bottom-up approach generating a tree-like structure known as a *dendogram*. A *dendogram* shows which clusters were merged in each layer of the tree, i.e. each layer in the dendogram is equivalent to an iteration representing a partitioning of the data set. The root node of the tree consists of one cluster and each leaf node of the tree represents a feature vector.

In the case of *divisive* hierarchical clustering, a top-down approach is followed where all feature vectors are initially assigned to a single cluster as the root node. In each iteration, clusters containing the most *dissimilar* feature vectors are split. The process continues until each feature vector represents a cluster as a leaf node in the *dendogram*.

There are different *linking* techniques to determine the two most *similar* clusters. Each of these techniques makes use of a proximity matrix containing the pairwise *similarities* between clusters [96]. Since the different types of *linking* techniques are applicable to both *divisive* and *agglomerative* hierarchical clustering, the remainder of this section focuses on *agglomerative* hierarchical clustering. The most popular and familiar *linking* techniques are [56, 96]:

- **Single link:** Also known as the *nearest-neighbour method* [56], the *similarity* between

two clusters is measured as the minimum distance between two feature vectors, one from each cluster, i.e.

$$\ell_{single}\left(C_i, C_j\right) = \min_{\forall \mathbf{p} \in C_i, \forall \mathbf{q} \in C_j} \{\sigma(\mathbf{p}, \mathbf{q})\} \tag{2.12}$$

where σ is a similarity measure, C_i and C_j are the i^{th} and j^{th} clusters respectively. A drawback of the single link technique is that the formed clusters are stretched out, i.e. a *chaining* effect [56, 96]. *Chaining* occurs when two clusters with highly dissimilar elements in each cluster are merged due to single elements being similar. Figure 2.1(a) illustrates the single link technique.

- **Complete link:** Also known as the *furthest-neighbour method* [56], the *similarity* between two clusters is measured as the maximum distance between two feature vectors, one from each cluster, i.e.

$$\ell_{complete}\left(C_i, C_j\right) = \max_{\forall \mathbf{p} \in C_i, \forall \mathbf{q} \in C_j} \{\sigma(\mathbf{p}, \mathbf{q})\} \tag{2.13}$$

  The complete link technique generates compact clusters [56, 96]. Figure 2.1(b) illustrates the complete link technique.

- **Average link:** The *similarity* between two clusters is measured as the average distance between all feature vectors from within the two clusters, i.e.

$$\ell_{average}\left(C_i, C_j\right) = \frac{1}{\left|C_i\right| \left|C_j\right|} \sum_{\forall \mathbf{p} \in C_i, \forall \mathbf{q} \in C_j} \sigma(\mathbf{p}, \mathbf{q}) \tag{2.14}$$

  The two clusters with the lowest $\ell_{average}$ value are merged into one cluster [56]. Figure 2.1(c) illustrates the average link technique.

- **Centroid link:** The distance between two centroids of different clusters can also measure the *similarity* between two clusters, i.e.

$$\ell_{centroid}\left(C_i, C_j\right) = \sigma\left(\mathbf{c}_i, \mathbf{c}_j\right) \tag{2.15}$$

  where $\mathbf{c}_i$ and $\mathbf{c}_j$ are the centroids of clusters C_i and C_j, respectively. The centroid of a cluster is defined in equation (2.18). The two clusters with the lowest $\ell_{centroid}$ value are merged into one cluster. Figure 2.1(d) illustrates the centroid link technique.

Hierarchical clustering algorithms do not have a pre-specified number of clusters. The number of clusters can be determined at any level of the *dendogram* or can be based on a *similarity* threshold [96]. Hierarchical clustering algorithms also make no assumption of the distribution of the

data set, i.e. the algorithms are independent of the initial conditions [56].

There are, however, a few drawbacks to the hierarchical approach to clustering data. Hierarchical clustering algorithms are not suitable to cluster data with overlapping clusters, or data that consists of clusters with varying shapes, sizes and/or densities [56]. Hierarchical clustering algorithms are also not suitable for very large data sets, since the proximity matrix of pairwise *similarities* does not scale well with large data sets. Once two clusters are merged, feature vectors assigned to a cluster cannot be re-assigned to a different cluster. Therefore hierarchical clustering algorithms are static and merged clusters cannot be separated [56].

2.3.2 Partitional Clustering

Partitional clustering algorithms *partition* feature vectors in a data set into a number of non-hierarchical clusters. Partitioning of these feature vectors optimises a specific objective function [96]. The objective function is optimised such that the *inter-cluster* distance is maximised and the *intra-cluster* distance minimised. The *inter-cluster* distance measures the average separation between the centroids of all possible pairs of clusters and is calculated as

$$J_{inter} = \frac{2}{K \times (K-1)} \sum_{k=1}^{K-1} \sum_{j=k+1}^{K} \sigma\left(\mathbf{c}_k, \mathbf{c}_j\right) \tag{2.16}$$

A larger J_{inter} value indicates a higher average separation between cluster centroids, whereas a smaller value indicates a lower separation between cluster centroids. The *intra-cluster* distance measures the *compactness* of the clusters and is calculated as

$$J_{intra} = \frac{\sum_{k=1}^{K} \sum_{\forall \mathbf{p} \in C_k} \sigma\left(\mathbf{p}, \mathbf{c}_k\right)}{|P|} \tag{2.17}$$

J_{intra} calculates the average of all the distances between each feature vector and the cluster centroid with which the feature vector is associated. Thus larger distances between the feature vectors and the associated cluster centroids will indicate less compact clusters and vice versa. Partitional clustering algorithms can be *exclusive*, *overlapping* or *probabilistic*. Each of these categories is explained next.

Exclusive Clustering: Also known as *crisp* or *hard* clustering, a feature vector is only grouped with a single cluster. The most familiar algorithm in this category is the iterative K-means clus-

tering algorithm [52]. K-means initialises K centroids, where K is the number of clusters into which a data set is partitioned. Based on a similarity measure, each feature vector in the data set is then assigned to only one of these centroids. A feature vector, $\mathbf{p}$, is assigned to a centroid, $\mathbf{c}$, if $\mathbf{p}$ is most similar to $\mathbf{c}$. Thus the subset of feature vectors assigned to a centroid forms a cluster.

After each feature vector in the data set is assigned to a centroid, the centroid of each cluster is recalculated according to the feature vectors assigned to the cluster. Algorithm 2.1 lists the pseudo code of a basic K-means algorithm [96].

Algorithm 2.1: Basic K-means

```
Randomly initialise K centroids;
while some stopping condition(s) not true do
    for each feature vector p_i ∈ P do
        Calculate the similarity between p_i and c_k, k = 1,...,K;
        Assign p_i to centroid c_k with which p_i has the highest similarity;
    end
    Recalculate the centroid of each cluster;
end
```

The *similarity* between a feature vector, $\mathbf{p}_i$, and a centroid, $\mathbf{c}_k$, is calculated using the Euclidean distance measure as defined in equation (2.3). Thus a lower value of σ implies a higher similarity. The centroid (mean), $\mathbf{c}_k$, of cluster, C_k is calculated as

$$\mathbf{c}_k = \frac{1}{|C_k|} \sum_{\forall \mathbf{p} \in C_k} \mathbf{p} \tag{2.18}$$

The K-means algorithm optimises the *sum of squared distances* [70] as objective function by minimising the *intra-cluster* distance. The *sum of squared distances* is defined as [96]

$$J_{SSE} = \sum_{k=1}^{K} \sum_{\forall \mathbf{p} \in C_k} \sigma(\mathbf{p}, \mathbf{c}_k)^2 \tag{2.19}$$

The J_{SSE} determines the clustering quality of the clustered data set.

The *stopping criteria* for K-means can be one of the following [26, 96]:

- when there is no change in the centroids,
- there is minimal reassignment of feature vectors to different centroids,

- the J_{SSE} is small enough or there is a minimal decrease in J_{SSE}, or
- a specified number of iterations have been reached.

Although K-means is a very simple clustering algorithm, it has a few drawbacks. Since K-means minimises the sum of squared errors, the algorithm is susceptible to *outliers* in a data set which inflate the J_{SSE} [81]. *Outliers* can be removed, but in cases where the data is dynamic, *outliers* might indicate a change in the data. *Outlier* analysis is discussed in section 2.6. Since the centroids are randomly initialised, each run of the K-means algorithm delivers different clustering results. Thus, the random initialisation of K cluster centroids also determines the clustering quality [17].

An enhancement to K-means is the bisecting K-means which is less susceptible to the initialisation of K centroids, since all feature vectors are initially grouped into one cluster [156]. Predicting the correct number of K clusters also influences the clustering quality [71]. The centroids can be initialised by randomly selecting K feature vectors from the data set. This is known as the K-medoids algorithm [107]. The most centrally located feature vector in a cluster is that cluster's *medoid*. Thus the objective of K-medoids is to find the optimal *medoids* in a data set.

Overlapping Clustering (also known as *fuzzy* clustering): A feature vector is grouped with all clusters to a certain degree of membership [188]. The most familiar algorithm in this category is the Fuzzy C-means clustering algorithm, which is explained next [16]. The Fuzzy C-means algorithm initialises a membership matrix, $\mathbf{M}^{I\times K}$, where I is the number of feature vectors in data set P, and K is the number of clusters (centroids) [56]. Thus, an element m_{ik} of matrix $\mathbf{M}$, is the degree of membership of a feature vector $\mathbf{p}_i$ to the centroid $\mathbf{c}_k$ of cluster C_k. m_{ik} satisfies the following constraints:

- $m_{ik} \in [0,1], \quad i = 1,\ldots,I \quad \text{and} \quad k = 1,\ldots,K;$
- $0 < \sum_{i=1}^{I} m_{ik} < I, \qquad k = 1,\ldots,K$, i.e. no empty clusters are allowed and no cluster may contain all feature vectors; and
- $\sum_{k=1}^{K} m_{ik} = 1, \qquad i = 1,\ldots,I.$

The degree of membership, m_{ik}, is defined as [56, 70]

$$m_{ik} = \frac{\left[\frac{1}{\sigma(\mathbf{p}_i,\mathbf{c}_k)^2}\right]^{\frac{1}{\phi-1}}}{\sum_{k=1}^{K}\left[\frac{1}{\sigma(\mathbf{p}_i,\mathbf{c}_k)^2}\right]^{\frac{1}{\phi-1}}} \tag{2.20}$$

where ϕ is the weighting exponent ($\phi \geq 1$) which controls the degree of *fuzziness* of the resulting clusters [56]. Thus, a higher value of ϕ increases the *fuzziness* of the algorithm. The centroid, $\mathbf{c}_k$, of cluster C_k is calculated as [56, 70]

$$\mathbf{c}_k = \frac{\sum_{i=1}^{I} (m_{ik})^{\phi} \mathbf{p}_i}{\sum_{i=1}^{I} (m_{ik})^{\phi}} \tag{2.21}$$

Algorithm 2.2 provides pseudo code for the Fuzzy C-means algorithm [56].

Algorithm 2.2: Fuzzy C-means

Randomly initialise K centroids;
Initialise matrix $\mathbf{M}$ by calculating m_{ik} as defined in equation (2.20);
repeat
  Recalculate the centroid of each cluster using equation (2.21);
  Update the degree of memberships m_{ik} with m'_{ik}, which is calculated using equation (2.20);
until $\max_{ik} \left\{ \left\| m_{ik} - m'_{ik} \right\| \right\} < \varepsilon$;

The objective function optimised by the Fuzzy C-means algorithm, is defined as [56, 70]

$$J_{FCM}(\mathbf{M}, C) = \sum_{i=1}^{I} \sum_{k=1}^{K} m_{ik}^{\phi} \sigma(\mathbf{p}_i, \mathbf{c}_k)^2 \tag{2.22}$$

where C is the set of K centroids and $\mathbf{M}$ is the matrix of membership degrees. Since Fuzzy C-means assigns a feature vector to a centroid with a certain degree of membership, the application of Fuzzy C-means is more realistic than K-means, because feature vectors tend to overlap. Similar to the K-means algorithm, the number of clusters needs to be specified. Fuzzy C-means may also converge to local optima [96].

Probabilistic Clustering: Probabilistic models assume that feature vectors are generated from different distributions, i.e. K clusters in a data set implies K different and unknown distributions in the data set [15, 56]. Thus, the data set consists of a mixture of density functions, one for each cluster [15, 21]. The probability density of the data is the sum of all the individual densities and is defined as [171]

$$G(\mathbf{p}; \Xi) = \sum_{k=1}^{K} \chi_k g(\mathbf{p}; \xi_k) \tag{2.23}$$

where g is a probability density function with parameters ξ_k. Ξ is the set of distribution parameters for each cluster, i.e. $\Xi = \{\xi_1, \ldots, \xi_k, \ldots, \xi_K\}$. Let g be the Gaussian density function, then $\xi_k = (\chi_k, \Omega_k, \mathbf{Z}_k)$ where Ω_k is the mean vector and $\mathbf{Z}_k$ the covariance matrix for the distribution of cluster k [15, 171]. The parameter χ in equation (2.23) is known as the *mixing* probability parameter [21, 171], and is the probability that feature vector $\mathbf{p}$ is generated from distribution k. Thus G is a mixture of Gaussian distributions with an unknown set of parameters, Ξ [21, 171]. The maximum likelihood method is a statistical technique to find Ξ [21]. Thus, the objective function that is optimised is defined as [21]

$$J_{EM}(\Xi) = \sum_{i=1}^{I} \log \left[\sum_{k=1}^{K} \chi_k g(\mathbf{p}_i; \xi_k) \right] \tag{2.24}$$

The above equation is optimised by the expectation maximisation (EM) algorithm [41]. The EM algorithm consists of an *expectation* step followed by a *maximisation* step in each iteration [21].

Algorithm 2.3 gives basic pseudo code for optimising equation (2.24) using EM. The algorithm stops when there is a small change in equation (2.24), which indicates that EM converged. There are a few drawbacks to the Gaussian Mixture model which are [56, 71]:

- the number of clusters needs to be specified,
- it is assumed that all clusters have a Gaussian distribution, and
- EM depends on the initial estimate of ξ.

2.3.3 Other Clustering Methods

Another clustering method is spectral clustering which is based on spectral graph theory [8, 143]. The patterns in a data set which need to be partitioned are represented as vertices and linked with weighted edges to form a connected graph. The connected graph can also be presented as a matrix of the distances between the patterns in the data set. The spectral clustering algorithm searches through the graph for edges which need to be pruned (or cut). The pruning of edges in the graph delivers a number of disjointed sub-graphs. Pruning is done in such a way that the similarities between vertices of the same sub-graph are higher than the pruned edge between two sub-graphs. The minimum-, ratio- or normalised-cut measures can be used to determine which edges need to be pruned [65, 100, 143]. Although spectral clustering can generate arbitrary-shaped clusters, there are two drawbacks to spectral clustering which are:

Algorithm 2.3: Basic Gaussian Mixture using EM

Set the number of iterations $t = 0$;

Estimate the initial values for $\xi_k^{(t)} = \left(\chi_k^{(t)}, \Omega_k^{(t)}, \mathbf{Z}_k^{(t)}\right)$;

repeat

Calculate the *expected* values of the unknown data (*expectation* step) using

$$\chi^{(t)}(k|\mathbf{p}_i) = \frac{\chi_k^{(t)} g\left(\mathbf{p}_i; \xi_k^{(t)}\right)}{\sum_{k=1}^{K} \chi_k^{(t)} g\left(\mathbf{p}_i; \xi_k^{(t)}\right)} \tag{2.25}$$

Calculate a new estimate for $\xi_k^{(t+1)}$ (*maximisation* step) using

$$\Omega_k^{(t+1)} = \frac{\sum_{i=1}^{I} \chi^{(t)}(k|\mathbf{p}_i)\, \mathbf{p}_i}{\sum_{i=1}^{I} \chi^{(t)}(k|\mathbf{p}_i)} \tag{2.26}$$

$$\mathbf{Z}_k^{(t+1)} = \frac{\sum_{i=1}^{I} \chi^{(t)}(k|\mathbf{p}_i) \left(\mathbf{p}_i - \Omega_k^{(t+1)}\right)^T \left(\mathbf{p}_i - \Omega_k^{(t+1)}\right)}{\sum_{i=1}^{I} \chi^{(t)}(k|\mathbf{p}_i)} \tag{2.27}$$

$$\chi_k^{(t+1)} = \frac{1}{I} \sum_{i=1}^{I} \chi^{(t)}(k|\mathbf{p}_i) \tag{2.28}$$

$t = t + 1$;

until $\left(J^{t+1} - J^t\right) < \varepsilon$;

- the method is computationally expensive, and
- the clustering performance is influenced by a user-specified kernel width parameter.

A common drawback of the discussed clustering methods in the previous section is that these methods have difficulty in identifying non-convex clusters. A solution to partitioning data with non-convex clusters is to change the set of feature vectors used to represent the data using a kernel method. A kernel function projects the feature vectors in a data set to a higher dimension where the feature vectors are linearly separable for partitioning. A well-known kernel-based clustering method is the Support Vector Machine (SVM). SVM is a binary classifier that constructs a linearly separating hyperplane between feature vectors of two classes. The hyperplane separates the feature vectors in such a way that the distance between the hyperplane and the feature vectors nearest to the hyperplane are maximised. SVM is repetitively executed for multi-class data sets. In the context of clustering data with non-convex clusters, the feature vectors in the

data set are transformed with a non-linear kernel function into a higher dimensional space which is linearly separable. SVM is then used to construct the hyperplanes (boundaries) between the transformed feature vectors. The initial feature vectors in the data set are then labelled according to the identified boundaries of the clusters in the data set.

2.4 Cluster Quality Validation

Since the identified number of groups (clusters) and the partitioning of data patterns between these groups may differ among different clustering algorithms, the quality of the partitioning needs to be evaluated, i.e. cluster validation quantitatively evaluates the clustering result of a clustering algorithm [170]. The different cluster validity measures are categorised into three criteria [67]:

- *internal* criteria - an example of this criteria is when a proximity matrix is used to evaluate the clustering results,
- *external* criteria - when an expected clustering result is pre-specified and the clustering results are evaluated against the expected clustering result, and
- *relative* criteria - clustering results are compared to other clustering schemes which are obtained by different input parameter values to the same algorithm.

A challenge in data clustering is to determine the optimal number of clusters in the data set. A drawback of the first two criteria to determine the optimal number of clusters is the statistical testing with high computational cost and the pre-specified clustering expectation. An approach to validate the number of clusters formed is to visually present the clustering results. In multi-dimensional problems where the number of dimensions is greater than three, visualisation of the formed clusters becomes difficult [67, 119].

Another approach to determine the optimal number of clusters is to execute the clustering algorithm multiple times, each time with a different number of clusters and validating the clustered data set with a cluster validity index, i.e. *relative* criteria. The cluster validity index is then plotted as a function of the number of clusters obtained for each execution of the algorithm. The number of clusters generated from the input parameters with the highest (or lowest) cluster validity index is then selected as the optimal number of clusters [151, 170]. This section discusses some of the most familiar cluster validity indices, which form part of the *relative* criteria to evaluate the partitioning quality of a clustering algorithm.

Dunn's index: The cluster validity index of Dunn [44] identifies clusters which are well separated and compact. Large values of the index imply well separated and compact clusters. Dunn's index is calculated as [66]

$$Q_D(K) = \min_{k=1,\ldots,K} \left\{ \min_{j=k+1,\ldots,K} \left\{ \frac{\sigma'(C_k, C_j)}{\max_{k=1,\ldots,K} \{\upsilon(C_k)\}} \right\} \right\} \tag{2.29}$$

where $\sigma'(C_k, C_j)$ is the dissimilarity between two clusters defined as

$$\sigma'(C_k, C_j) = \ell_{single}(C_k, C_j) \tag{2.30}$$

and $\upsilon(C_k)$ is the diameter of cluster C_k defined as

$$\upsilon(C_k) = \max_{\forall \mathbf{p},\mathbf{q} \in C_k} \{\sigma(\mathbf{p}, \mathbf{q})\} \tag{2.31}$$

where σ is the Euclidean distance as defined in equation (2.3) and ℓ_{single} is defined in equation (2.12). A data set with well-separated clusters has large inter-cluster distances as well as small intra-cluster distances for compact clusters. Thus, from the above Dunn-index definition, inter-cluster distances, σ', are maximised and intra-cluster distances, υ, minimised to maximise the value of Q_D. The maximum Q_D index value for a specific value of K indicates the optimal clustering of the data set. Problems with the Q_D index listed in [66] are that it is

- computationally complex, and
- sensitive to noise in the data set (noise increases the value of υ).

Net Information Gain index (NIG): An enhancement to the Dunn-index is the net information gain (NIG) validity index [103]. NIG measures the information change between clusters when a new cluster is introduced. NIG is applicable to clustering algorithms which are executed multiple times to determine the optimal number of clusters [103]. Initially all feature vectors in the first execution of the algorithm form a single cluster, i.e. execution E_1 has one cluster, C_1. With each execution of the algorithm, E_i, the data set, P, is re-clustered into i clusters using a clustering algorithm like K-means, i.e. $P = \cup_{k=1}^{i} C_k^i$. The *migration* of feature vectors between clusters from execution E_i to E_{i+1} forms the base for cluster quality measurement using NIG. Three different feature vector *migration* types are defined in [103] and discussed next. Let C_k^i be the k^{th} cluster in execution E_i, i.e. $1 \leq k \leq i$. Then, migration types are defined as:

1. *stagflation* - feature vectors forming a single cluster C_k^i in execution E_i continue to be part of that cluster in execution E_{i+1}, i.e. $C_j^{i+1} = C_k^i$ where $1 \leq j \leq (i+1)$.

2. *leakage* - a few feature vectors forming part of cluster C_k^i in execution E_i can be grouped with other feature vectors to form a different cluster C_j^{i+1} in execution E_{i+1} where $1 \leq j \leq (i+1)$.

3. *disassociation* - feature vectors forming part of a cluster C_k^i in execution E_i, divide into two or more smaller clusters in execution E_{i+1}, i.e. $\cup_{j=1}^{J} C_j^{i+1} \subseteq C_k^i$, where J is the number of clusters evolved from cluster C_k^i and $2 \leq J \leq (i+1)$.

When a cluster C_k^i divides into more than two clusters, the two most dominant clusters in execution E_{i+1} are selected to calculate the information change on cluster C_k^i. The two most dominant clusters are those clusters containing the most and second most number of migrated feature vectors from cluster C_k^i, respectively. The information gain/loss on cluster C_k^i from execution E_i to execution E_{i+1} is calculated as

$$inf\left(C_k^i\right) = d\left(C_k^i\right) \times M\left(C_k^i\right) \tag{2.32}$$

where $d\left(C_k^i\right)$ is the direction of the magnitude of change in information. The magnitude of change, $M\left(C_k^i\right)$, measures the *migration* of feature vectors from cluster C_k^i, defined as [103]

$$M\left(C_k^i\right) = -\sum_{j=1}^{J} p_j \ln p_j \tag{2.33}$$

where J is the number of clusters to where feature vectors of cluster C_k^i migrate to and p_j is the fraction of feature vectors migrating from cluster C_k^i to cluster C_j^{i+1}. If migrated feature vectors in cluster C_j^{i+1} overlap with feature vectors in cluster C_k^i then the direction of change $d\left(C_k^i\right) = -1$, i.e. *information loss*. If migrated feature vectors in cluster C_j^{i+1} are well separated from patterns in cluster C_k^i then the direction of change $d\left(C_k^i\right) = 1$, i.e. *information gain*. The centroid diameter and centroid linkage are used to measure the overlap between clusters [103]. The centroid diameter of a cluster C_k is defined as [103]

$$\upsilon\left(C_k\right) = 2\left[\frac{\sum_{\forall \mathbf{p} \in C_k} \sigma\left(\mathbf{p}, \mathbf{c}_k\right)}{|C_k|}\right] \tag{2.34}$$

where $|C_k|$ is the number of feature vectors in cluster C_k, $\mathbf{c}_k$ is the centroid of cluster C_k, and σ is a distance measure. The centroid linkage between two clusters C_k and C_j is defined in equation (2.15) [103]. The direction of change for cluster C_k^i from execution E_i to execution E_{i+1} is defined as

$$d\left(C_k^i\right) = \begin{cases} 1 & \text{if} \quad \ell_{centroid}\left(C_k^i, C_j^{i+1}\right) \geq \frac{1}{2}\left[\upsilon\left(C_k^i\right) + \upsilon\left(C_j^{i+1}\right)\right] \\ -1 & \text{otherwise} \end{cases} \tag{2.35}$$

The net information gain between two executions E_i and E_{i+1} is calculated as

$$NIG_{i+1} = \sum_{k=1}^{i} inf\left(C_k^i\right) \tag{2.36}$$

The total information content of the i^{th} execution is calculated as the cumulative sum of NIG's over all executions prior to and including execution i, defined as

$$Q_{NIG} = \sum_{g=0}^{i} NIG_g \tag{2.37}$$

The execution of the algorithm with the largest Q_{NIG} index value is considered as the execution with optimal clustering.

C-index: Let $\mathcal{E}$ be the ascending ordered set of distances between all possible pairs of feature vectors in data set P, i.e. $|\mathcal{E}| = \frac{|P|\times(|P|-1)}{2}$. Let $\mathcal{S}$ be the sum of m feature vector pair distances, where each feature vector pair is of the same cluster, i.e. $\mathbf{p},\mathbf{q} \in C_k$ where $\mathbf{p}$ and $\mathbf{q}$ are a pair in cluster C_k such that $\mathbf{p} \neq \mathbf{q}$. Then the C-index [88] is calculated as [19]

$$Q_C = \frac{\mathcal{S} - \mathcal{S}_{min}}{\mathcal{S}_{max} - \mathcal{S}_{min}} \tag{2.38}$$

In the above definition of Q_C, $\mathcal{S}_{max}$ and $\mathcal{S}_{min}$ are defined as

$$\mathcal{S}_{min} = \sum_{i=1}^{m} e_i \tag{2.39}$$

$$\mathcal{S}_{max} = \sum_{i=|\mathcal{E}|-m+1}^{|\mathcal{E}|} e_i \tag{2.40}$$

where $e_i \in \mathcal{E}$ and $m \geq 1$. Thus, S_{min} and S_{max} are the sum of the m smallest and m largest distances between feature vector pairs in P, respectively. The denominator in the definition of Q_C normalises the index value such that $Q_C \in [0,1]$. Smaller values of Q_C imply clusters of better quality. The optimal number of clusters minimises the Q_C index. The C-index is a suitable validity index for clusters of similar sizes.

Davies-Bouldin index: Davies and Bouldin (DB) proposed an index that measures the average similarity between each cluster and the cluster most similar to it [31]. The DB-index is calculated as [68]

$$Q_{DB} = \frac{1}{K}\sum_{k=1}^{K} \max_{\substack{j=1,\ldots,K \\ j\neq k}} \left\{ \frac{\frac{1}{2}\upsilon(C_k) + \frac{1}{2}\upsilon(C_j)}{\sigma(\mathbf{c}_k, \mathbf{c}_j)} \right\} \tag{2.41}$$

where K is the number of clusters, σ is the Euclidean distance as defined in equation (2.3), υ is the cluster centroid diameter as defined in equation (2.34), and $Q_{DB} \in [0,\infty)$. In the above definition, Q_{DB} has a small value when the distance between centroids $\mathbf{c}_k$ and $\mathbf{c}_j$ is large and the corresponding clusters C_k and C_j of these centroids are compact. Thus, an optimal number of K clusters minimises the value of Q_{DB}.

Halkidi-Vazirgiannis index: The *S_Dbw* index proposed by Halkidi and Vazirgiannis is calculated as [69]

$$Q_{S\_Dbw}(K) = Scat(K) + Dens\_bw(K) \tag{2.42}$$

The *S_Dbw*-index is defined as the summation of the average scattering (compactness, i.e. *intra-cluster variance*) of the clusters and the density among the clusters (separation, i.e. *inter-cluster density*). $Scat(K)$ is the average scattering of K clusters and $Dens\_bw(K)$ is the density among the K clusters. $Scat(K)$ is defined as [69]

$$Scat(K) = \frac{1}{K}\sum_{k=1}^{K} \frac{\psi(C_k, \mathbf{c}_k)}{\psi(P, \mathbf{p})} \tag{2.43}$$

where $\mathbf{p}$ is the centroid of data set P and ψ is the variance of a set of feature vectors, defined as [69]

$$\psi(V, \mathbf{v}_j) = \sqrt{\sum_{n=1}^{N} \left[\frac{1}{|V|} \sum_{i=1}^{|V|} (\mathbf{x}_{i,n} - \mathbf{v}_{j,n})^2 \right]^2} \tag{2.44}$$

where N is the dimensionality of feature vectors, $\mathbf{x} \in V$. Therefore, the average standard deviation of all clusters is defined as [69]

$$\iota = \frac{1}{K}\sqrt{\sum_{k=1}^{K} \psi(C_k, \mathbf{c}_k)} \tag{2.45}$$

$Dens\_bw(K)$ is defined as [69]

$$Dens\_bw(K) = \frac{1}{K \times (K-1)} \sum_{k=1}^{K} \sum_{\substack{j=1 \\ k \neq j}}^{K} \frac{density(\mathbf{u}_{kj})}{\max\{density(\mathbf{c}_k), density(\mathbf{c}_j)\}} \tag{2.46}$$

where $\mathbf{u}_{kj}$ is the middle point of cluster centroids $\mathbf{c}_k$ and $\mathbf{c}_j$, and is calculated as $\mathbf{u}_{kj} = \frac{\mathbf{c}_k + \mathbf{c}_j}{2}$. The $density(\mathbf{u}_{kj})$ of a feature vector $\mathbf{u}_{kj}$ calculates the number of feature vectors in the neighbourhood of vector $\mathbf{u}_{kj}$. In order to determine whether a feature vector is within the neighbourhood of another vector, the following neighbourhood function is defined [69]

$$n(\mathbf{p}_i, \mathbf{u}) = \begin{cases} 0 & \text{if} \quad \sigma(\mathbf{p}_i, \mathbf{u}) > \iota \\ 1 & \text{otherwise} \end{cases} \tag{2.47}$$

Thus, a feature vector $\mathbf{p}_i$ is within the neighbourhood of $\mathbf{u}$ if the distance from $\mathbf{u}$ is less than ι which is the average standard deviation of all clusters as defined in equation (2.45); ι is the radius of the neighbourhood. The $density(\mathbf{u}_{kj})$ is then calculated as [69]

$$density(\mathbf{u}_{kj}) = \sum_{\forall \mathbf{p}_i \in \{C_k \cup C_j\}} n(\mathbf{p}_i, \mathbf{u}_{kj}) \tag{2.48}$$

A small value of $Scat(K)$ indicates compact clusters and a small value of $Dens\_bw(K)$ indicates well separated clusters [69]. Thus the number of clusters, K, that minimises the Q_{S_Dbw} index value is considered as the optimal number of clusters in the data set.

Ray-Turi index: Ray and Turi proposed a validity index which is based on the ratio of *intra-clustering* distance to *inter-clustering* distance [151]. The proposed index is calculated as [151]

$$Q_{ratio} = \frac{J_{intra}}{inter_{min}} \tag{2.49}$$

where J_{intra} is defined in equation (2.17), $inter_{min}$ is calculated as

$$inter_{min} = \min_{\substack{k=1,\ldots,K-1 \\ j=k+1,\ldots,K}} \left\{ \sigma\left(\mathbf{c}_k, \mathbf{c}_j\right) \right\} \tag{2.50}$$

and σ is the Euclidean distance as defined in equation (2.3). In the above definition of *intra*, the average *compactness* of the clusters is calculated by averaging over all the distances between each cluster's centroid and the feature vectors within that cluster. The definition of $inter_{min}$ simply calculates the smallest distance between the centroids of the clusters to determine the smallest separation between clusters. J_{intra} needs to be minimised and $inter_{min}$ needs to be maximised for more compact and more separated clusters. Thus, the defined ratio validity index, Q_{ratio}, needs to be minimised to have optimal clustering. Therefore the optimal number of clusters, K, minimises the value of Q_{ratio}.

Turi proposed a modification to the above ratio of *intra-clustering* distance to *inter-clustering* distance by multiplying the ratio with a Gaussian function of the number of clusters [173]. The modified index is calculated as [173]

$$Q_{RT} = Q_{ratio} \times \left[c \times g\left(\mu, \sigma\right) + 1\right] \tag{2.51}$$

where g is a Gaussian function with mean, μ, and standard deviation, σ and c is some constant. Function g is defined as

$$g\left(\mu, \sigma\right) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left[-\frac{(K-\mu)^2}{2\sigma^2}\right]} \tag{2.52}$$

where K is the number of clusters. The Gaussian function penalises the ratio for small values of K in favour of larger values of K.

Other familiar cluster validity indices not covered in this section are among others Entropy, Purity and Silhouette. The interested reader is referred to [189] for more information.

2.5 Cluster Quality in Dynamic and Uncertain Environments

Section 2.4 gave an overview of different cluster quality measurements to quantitatively evaluate the clustering results of a clustering algorithm applied to stationary data sets, i.e. static environments. A static environment is defined as feature vectors in space which do not move to different

spatial positions over time, i.e. the feature vectors are static and will remain at the same positions at any given point in time. The cluster validity indices form part of the *relative* criteria.

This section discusses different types of non-stationary environments and introduces the different performance measures applied to optimisation algorithms for problems in non-stationary environments, i.e. dynamic environments. These performance measures are used to quantify the quality of partitioning by clustering algorithms in dynamic environments (discussed in section 7.1). A dynamic environment in the context of this thesis is defined as feature vectors in space which move or adapt to different spatial positions over time [64].

The goal of optimisation algorithms (such as particle swarm optimisation (PSO) which is discussed in section 2.7.1) is to locate an optimum to an optimisation problem. There are different classes of optimisation algorithms [162]. For the purpose of this section, stochastic population based optimisation algorithms are considered where a population of candidate solutions, $A(t)$, is maintained at each time step, t. The fitness of each candidate solution, $\mathbf{a} \in A(t)$, is calculated using the objective function, f, that needs to be minimised or maximised. The candidate solution with the *best* fitness, $best(t)$, is selected as the solution that best optimises the objective function, f, at a specific time step, t. An objective function can also change over time. These changes result in a dynamic search space with different optima at each point in time. Optimisation algorithms for dynamic environments need to track optima over time by detecting and tracking changes in the search space. The remainder of this section assumes maximisation of the objective function.

Changes in a dynamic environment can occur at any point in time with different effects to the optima of the objective function. Figure 2.2 illustrates the different types of dynamic environments for the following dynamic function:

$$f(\mathbf{x}, \boldsymbol{\omega}(t)) = \begin{cases} \frac{\omega_1(t)}{8} \times \pi \times \exp\left[-\frac{1}{2} \times \left((x_1 + \omega_2(t))^2 + (x_2 + \omega_2(t))^2\right)\right] & \text{if } (x_1 < 0) \text{ and } (x_2 < 0) \\ \frac{\omega_1(t)}{4} \times \pi \times \exp\left[-\frac{1}{2} \times \left((x_1 - \omega_2(t))^2 + (x_2 + \omega_2(t))^2\right)\right] & \text{if } (x_1 > 0) \text{ and } (x_2 < 0) \\ \frac{\omega_3(t)}{4} \times \pi \times \exp\left[-\frac{1}{2} \times \left(x_1^2 + (x_2 - 5)^2\right)\right] & \text{if } (x_2 > 0) \\ 0 \quad \text{otherwise} \end{cases} \tag{2.53}$$

where $\omega(t)$ is the control parameters which determine the magnitude of change in the dynamic environment at a specific time t. The different types of dynamic environments in [101, 180] are grouped into three main types [46]:

1. The locations of the optima change but the values of the optima remain the same (as illustrated in figure 2.2(b)).

2. The locations of the optima remain the same (no change) but the values of the optima change (as illustrated in figure 2.2(d)).

3. Both the locations and values of the optima change (as illustrated in figures 2.2(c) and 2.2(e)).

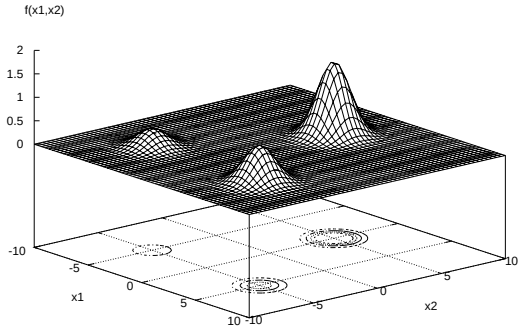
For each of these dynamic environment types the number of optima may change, in that new optima may appear and existing optima may disappear.

The cluster validity indices discussed in section 2.4 are based on two functions, namely *inter-* and *intra-error* (as defined in equations (2.16) and (2.17), respectively). The *inter-error* function needs to be maximised to obtain well separated clusters and the *intra-error* function needs to be minimised to obtain more compact clusters. Since the different cluster validity indices discussed in section 2.4 are either maximised or minimised to obtain the best partitioning of a data set, these indices can be used as fitness functions to achieve optimal clustering in dynamic environments.

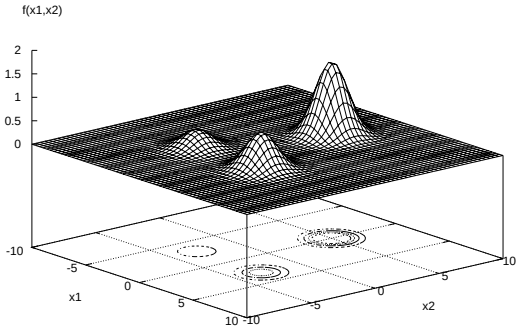
Thus, from a clustering perspective in a dynamic environment, the initial formed clusters of a data set can *adapt* over time, which means that at each time step the feature vectors in different clusters can follow different *migration* types to and from other clusters. These *migration* types were defined in section 2.4 as part of the discussion on the net information gain index (Q_{NIG}) [103]. The *migration* of feature vectors from one cluster to another implies that the centroids of the different clusters also move in space to different positions. Therefore centroids may move, disappear and/or new centroids may appear.

From the above list of dynamic environment types, the definition of clustering in section 2.1 and the different *migration* types discussed in section 2.4, the dynamic environments investigated in this thesis are defined as adapting feature vectors such that:

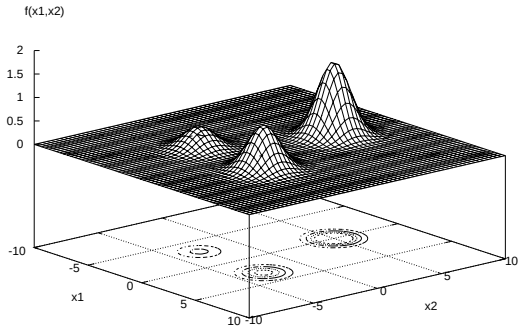
- the number of clusters remains static with the same centroids but the *compactness* of each cluster changes, i.e. movement of feature vectors within a static number of clusters (as illustrated in figure 2.3(b)).



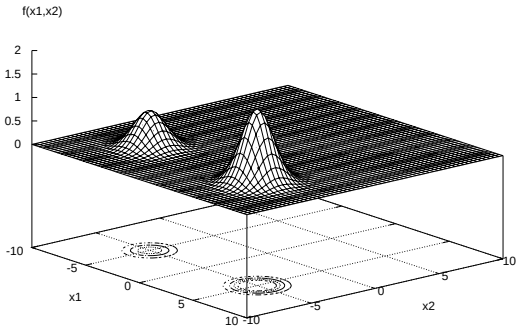
(a) Static function $\omega_1 = 1, \omega_2 = 5, \omega_3 = 2$



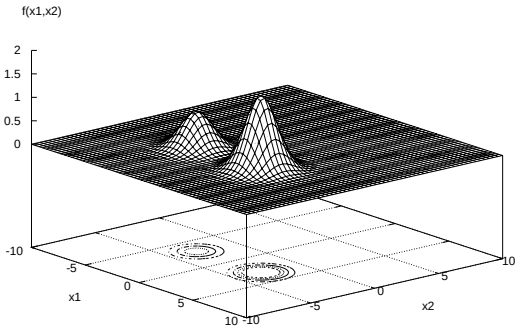
(b) Dynamic function $\omega_1 = 1, \omega_2 = 3, \omega_3 = 2$



(c) Dynamic function $\omega_1 = 1.2, \omega_2 = 3, \omega_3 = 2$

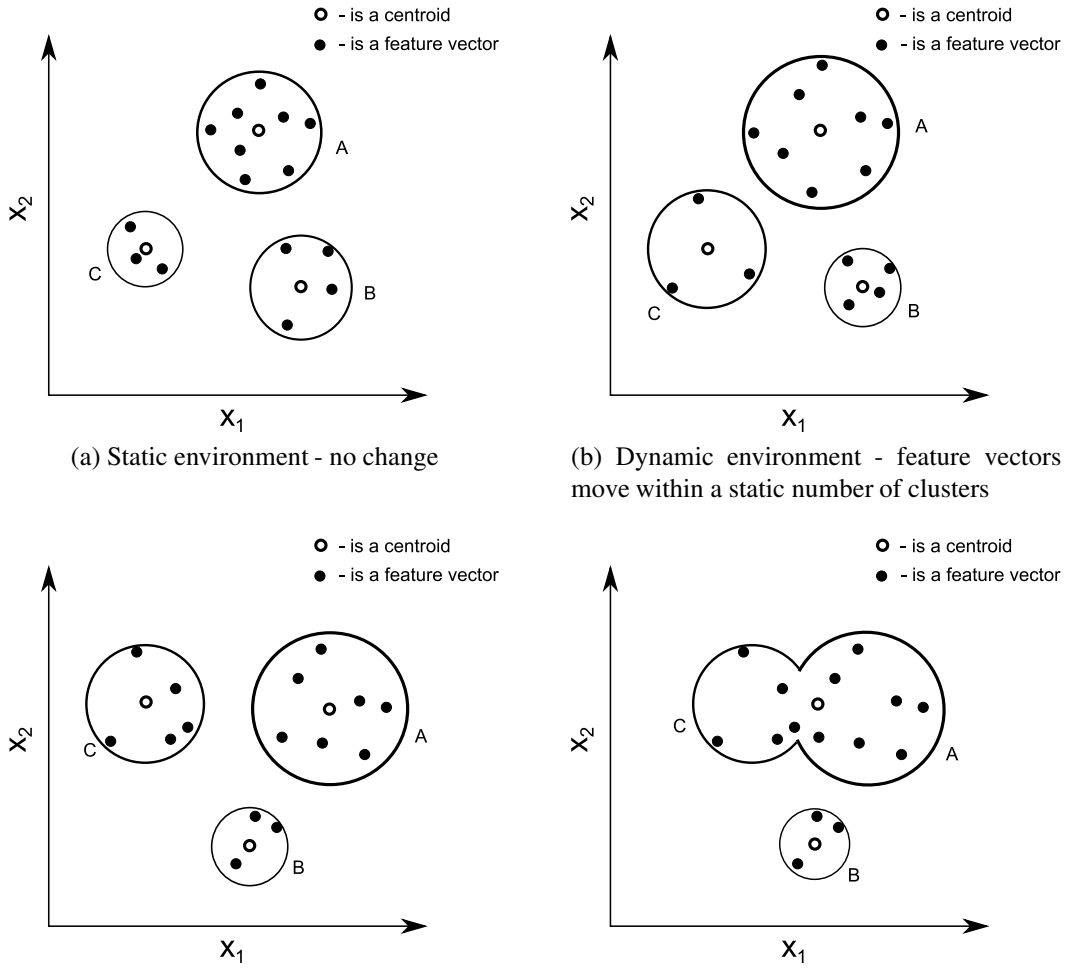


(d) Dynamic function $\omega_1 = 2, \omega_2 = 5, \omega_3 = 0$



(e) Dynamic function $\omega_1 = 2, \omega_2 = 3, \omega_3 = 0$

Figure 2.2 Dynamic Objective Function (equation (2.53))



(a) Static environment - no change

(b) Dynamic environment - feature vectors move within a static number of clusters

(c) Dynamic environment - a static number of centroids move with migration of feature vectors between clusters

(d) Dynamic environment - moving centroids with merging/dividing clusters and migrating feature vectors

Figure 2.3 Clustering in Dynamic Environments

- the number of clusters remains static with changing centroids and the *compactness* of each cluster changes, i.e. movement of a static number of clusters as well as *migration* of feature vectors (as illustrated in figure 2.3(c)).
- the number of clusters changes, resulting in different centroids and a change in the *compactness* of each cluster, i.e. clusters *merge/divide* as a result of feature vectors *migrating* between clusters and/or moving centroids (as illustrated in figure 2.3(d)).

The remainder of this section discusses the different performance measures of an optimisation algorithm which inspired the definition of a performance measure for clustering algorithms in dynamic environments (discussed in section 7.1). Performance measures that can be used to

quantify different aspects of the performance of optimisation algorithms at specific time intervals include (assuming maximisation of the objective function):

- **Accuracy:** This performance measure calculates the instantaneous accuracy of the best solution found by an optimisation algorithm at a certain time step, t. Feng [50] introduced an accuracy measure in static environments which can be calculated in dynamic environments as [180]

$$accuracy(t) = \frac{f(best(t)) - f_{min}(t)}{f_{max}(t) - f_{min}(t)} \tag{2.54}$$

  where f is the fitness function, $best(t)$ is the best candidate solution in the population at time step t and $f_{max}(t)$ and $f_{min}(t)$ are the maximum and minimum values of f in the search space at time step t, respectively, defined as:

$$f_{max}(t) = \max_{\forall \mathbf{x}(t) \in \Re^N} \{f(\mathbf{x}(t))\} \tag{2.55}$$

$$f_{min}(t) = \min_{\forall \mathbf{x}(t) \in \Re^N} \{f(\mathbf{x}(t))\} \tag{2.56}$$

  Note that $accuracy(t) \in [0,1]$, where an accuracy of one is the best possible value.

- **Stability:** An optimisation algorithm is defined to be *stable* if changes in the environment have a minor or no affect on the measured accuracy [180]. The stability of an optimisation algorithm at a certain time step, t, is calculated as [180]

$$stab(t) = \max\{0, accuracy(t-1) - accuracy(t)\} \tag{2.57}$$

  where $stab(t) \in [0,1]$. A $stab(t)$ value close to zero implies a high stability. If $stab(t) > 0$, then there is a difference in the accuracy between consecutive time steps. In the above calculation of *stability* (assuming maximisation), whenever $accuracy(t) > accuracy(t-1)$, the result is $stab(t) = 0$ which indicates that the optimisation algorithm is *stable*. This, however, is not true since there was an effect on the accuracy. The only time a $stab(t)$ value will have a deviation from zero is when the measured accuracy decreases (in the case of maximising the objective function). Whenever the measured accuracy improves due to a change in the environment, the $stab(t)$ value will remain zero and therefore give no indication of any change in the environment or the ability of the optimisation algorithm to track a moving optimum. Instead of measuring the *stability* of an optimisation algorithm, this thesis proposes that the *sensitivity* of the optimisation algorithm to a change in

the environment can be measured using

$$sens(t) = |accuracy(t-1) - accuracy(t)| \tag{2.58}$$

where $sens(t) \in [0,1]$. A $sens(t)$ value close to zero indicates a less *sensitive* change in the environment whereas a $sens(t)$ value closer to one indicates a more *sensitive* change. For both $stab(t)$ and $sens(t)$, the calculated values give an indication of the ability of the optimisation algorithm to track a moving optimum.

- **Recovery:** An optimisation algorithm also needs to react to a changing environment [180]. The reaction of an optimisation algorithm to a change in the environment at time t is measured as the time taken to locate the moved optimum at time $t^{'}$. An optimisation algorithm succeeds in the location of the moved optimum when the ratio of the accuracy at time $t^{'}$ to the accuracy at time t is greater or equal to the specified accuracy threshold. The accuracy threshold is calculated as $1-\varepsilon$ where ε is the minimum ratio of accuracy for an optimisation algorithm to succeed in locating the moved optimum. The ε-*reactivity* of an optimisation algorithm at time t is calculated as [180]

$$react_{\varepsilon}(t) = \begin{cases} t^{'} - t | t < t^{'} \leq T, \quad t^{'} \in \mathbb{N} & \text{if } \frac{accuracy\left(t^{'}\right)}{accuracy(t)} \geq (1-\varepsilon) \\ T - t & \text{otherwise} \end{cases} \tag{2.59}$$

where T is the total number of time steps and $\varepsilon \in [0,1]$. A smaller $react_{\varepsilon}$ value implies a higher reactivity.

A drawback to the above performance measures is that the maximum and minimum values of f at each time step t in the dynamic search space need to be known to determine accuracy, stability and reactivity [180]. The maximum and minimum fitness values might also change over time, due to the dynamic behaviour of the search space. This implies that the best fitness value at time t, might be the worst fitness value at time $t+1$ [131]. Thus, prior *knowledge* of the dynamic search space at each time step needs to be known to calculate the accuracy of the best solution found by an optimisation algorithm.

In cases where there is *no knowledge* of the dynamic search space, the above accuracy measure is inappropriate. In absence of information about the search space, the following accuracy measures can be used (assuming maximisation) [180]:

Current best fitness: The current best fitness is the most familiar accuracy measure for optimisation algorithms applied to dynamic environments, calculated as [180]

$$currentBest\left(t\right) = \max_{\forall \mathbf{a} \in A(t)} \left\{ f\left(\mathbf{a}\right) \right\} \tag{2.60}$$

where $f\left(\mathbf{a}\right)$ is the fitness of solution $\mathbf{a}$ in the population of candidate solutions A at time t. The *currentBest* performance measure calculates the fitness of the solution that best optimises the objective function at each time step.

Current best offline fitness: The current best offline fitness accuracy measure calculates the maximum current best accuracy up to a certain point in time [131, 180]. Thus, it measures the solution that best optimises the objective function over all time steps. The current best offline fitness accuracy measure is calculated as [180]

$$currentBestOff\left(t\right) = \max_{1 \leq t' \leq t} \left\{ currentBest\left(t'\right) \right\} \tag{2.61}$$

where *currentBest* is defined in equation (2.60). The $currentBestOff$ is less suitable in dynamic environments since it irrationally compares *currentBest* measures at different steps in time at which a change in the environment could occur. Thus, measuring the accuracy of an optimisation algorithm over a period of time as the solution that best optimises the objective function at a certain time within that period, has no meaning in a dynamic environment.

Current average fitness: The current average fitness calculates the average accuracy of population A at time t, defined as [102, 180]

$$currentAvg\left(t\right) = \frac{1}{\left|A\left(t\right)\right|} \sum_{\forall \mathbf{a} \in A(t)} f\left(\mathbf{a}\right) \tag{2.62}$$

A drawback to the current average fitness accuracy measure is that if some of the solutions in population A diverge from the optimal solution due to a change in the environment, the average accuracy of population A will decrease even though other solutions in population A converge to the new optimum. In such a case, the current average fitness accuracy measure gives a deceptive view of the optimisation algorithm's ability (or inability) to track moving optima.

Window accuracy: The window accuracy assumes that the *best* fitness does not change within a certain time-span, thus the accuracy is only measured within a certain window size, W [131,

180]. The window accuracy measure is calculated as [180]

$$windowAcc(t) = \max_{\forall \mathbf{a} \in A(t)} \left\{ \frac{f(\mathbf{a}) - windowWorst}{windowBest - windowWorst} \right\} \tag{2.63}$$

where

$$windowBest = \max_{t-W \leq t' \leq t} \left\{ \max_{\forall \mathbf{a} \in A(t')} \{f(\mathbf{a})\} \right\} \tag{2.64}$$

$$windowWorst = \min_{t-W \leq t' \leq t} \left\{ \min_{\forall \mathbf{a} \in A(t')} \{f(\mathbf{a})\} \right\} \tag{2.65}$$

The assumption of no change of the *best* fitness is a risk and a disadvantage of the *windowAcc* measure [131, 180].

The measured performance of different optimisation algorithms can be compared by averaging each algorithm's performance measure at each step in time over the total running time T [102]. This will give each algorithm a mean value of measured performance which is used for comparison. An accuracy measure related to the above *currentBest* measure is proposed in [131], namely the collective mean fitness (cmf). The collective mean fitness takes the fitness trajectory across the entire dynamic landscape into account by averaging the mean value of measured performance over the number of independent runs, E, of the algorithm on the same problem. The collective mean fitness is defined as [131]

$$cmf = \frac{\sum_{r=1}^{E} \left(\frac{\sum_{t=1}^{T} f(best(t))}{T} \right)}{E} \tag{2.66}$$

where T is the total number of time steps, and $f(best(t))$ is the *best* fitness value at time t. The cmf is thus the sum of all average *best* fitness values, averaged over a number of runs. The number of time steps T needed by an optimisation algorithm is important in optimisation of a dynamic environment. A too small value of T, will result in an unstable value of cmf. A large value of T is necessary for the optimisation algorithm to be exposed to extreme changes in the dynamic environment [131].

Section 7.1 proposes a clustering performance measure to quantify the quality of partitioning by clustering algorithms in dynamic environments. The proposed clustering performance measure is derived from the above collective mean fitness measure and uses cluster validity indices.

As mentioned earlier, the cluster validity indices are based on the *inter-* and *intra-errors* to determine the cluster separation and cluster compactness, respectively. In the context of clustering of dynamic environments, these *errors* can be used, in addition to the proposed clustering performance measure in section 7.1, to quantify the quality of partitioning by clustering algorithms over time.

As an example, assume a fixed number of clusters, K. The average *inter-error* plotted against time, will increase in value if the clusters become more separated in time, i.e. clusters move away from one another. If there is any *migration* of feature vectors between clusters, it is expected that the average *intra-error* plotted against time will fluctuate from the time of *migration* until the feature vectors become stationary again. In clustering problems where K is not fixed, the validity indices determine the optimal partitioning into K clusters at a specific time, indicating whether a new cluster emerged or whether clusters merged.

2.6 Outlier Detection and Analysis

Referring to the definition of data clustering in section 2.1, each cluster (or centroid) represents a *concept* or *trend* in the data set. Based on a *similarity* measure, an *outlier* feature vector is either not grouped with any cluster or has a major deviation from the centroid of a cluster with which the *outlier* is associated. Therefore an *outlier* is also known as an *exception* and is defined as a vector which is not *similar* to any of the centroids. *Outliers* are grossly different from and/or inconsistent with feature vectors of the same data set [74], which can be a result of inherent data variability [74].

In the context of a dynamic environment a feature vector can only be classified as an *outlier* at a specific point in time. An *outlier* at time t might disappear at time $t+1$ or even more *outliers* might occur within the same area. The latter could indicate a new trend in the data for a certain period of time.

Outlier detection and analysis is referred to as *outlier mining* and is described as follows [186]: In a data set of I feature vectors, the expected number of outlier vectors, o, are those feature vectors which are the most *dissimilar*, *exceptional* and/or *inconsistent* compared to the remainder of the data set. Outlier detection can be categorised into three approaches, namely the statistical approach, distance-based approach and deviation-based approach [74, 186]. Each of these

categories is briefly explained next.

Statistical approach: Feature vectors in a data set are identified as *outliers* with a *statistical discordancy* test by examining a *null* hypothesis and an *alternative* hypothesis. A *null* hypothesis can state that all feature vectors in a data set are from the same distribution. Thus, the *statistical discordancy* test verifies whether a feature vector is significantly large in relation to the distribution of the data set [74]. The *null* hypothesis is kept in case no statistical significant evidence supports the rejection thereof.

The *alternative* hypothesis states that a feature vector comes from a different distribution as the one defined in the *null* hypothesis [74]. The interested reader is referred to [74] for more information on the different *alternative* distributions.

A drawback of the statistical approach is the assumption of a specific distribution (like normal, Poisson etc.) for the data set and thus requires knowledge of the distribution parameters (like average, standard deviation etc.) and the expected number of outliers. Another major drawback is that the hypothesis testing is for outlier detection of single *features*, i.e. only a single attribute is tested in a feature vector. There is also no guarantee that all outliers are detected [74].

Distance-based approach: This approach is based on a distance measure between feature vectors in a data set. A global distance-based neighbourhood radius is defined for each feature vector, $\mathbf{p}_i$. If a fraction of the feature vectors is not within the distance radius of $\mathbf{p}_i$, then $\mathbf{p}_i$ is detected as an outlier in the data set [117]. The different distance-based outlier detection algorithms are the index-based, nested-loop and cell-based algorithms [74, 117]. A drawback to these distance-based algorithms is the user specified parameters of neighbourhood radius and the number of feature vectors in the data set that needs to be within the specified radius.

Deviation-based approach: The most general *concepts* can be derived from the feature vectors in a data set. Feature vectors which deviate from these general *concepts* are seen as outliers. There are two techniques in deviation-based outlier detection [74], namely sequential exception and the on-line analytical processing (OLAP) data cube technique. The first of these two techniques is discussed next and the interested reader is referred to [74] for more information on the OLAP technique:

- *Sequential exception technique:* The sequential exception technique is based on a process followed by humans to detect an outlier after being represented with a series of *similar* feature vectors [5]. An outlier is defined as a feature vector that deviates from the series.

  A sequence of subsets, $\{S_1, S_2, \ldots, S_o\}$, is built from a data set, P, consisting of I feature vectors, i.e. $2 \leq o \leq I$. Thus, $S_{o-1} \subset S_o : S_o \subseteq S$. A function of *dissimilarity* (not necessarily distance based) is calculated between each subset. The *dissimilarity* function is defined as any function that returns a low value to indicate more *similar* feature vectors and a high value to indicate less *similar* feature vectors [74, 186]. An example of a *dissimilarity* function is defined in equation (2.44), which calculates the variance of a set of feature vectors.

  A *smoothing factor* function is calculated for each subset, S_o, in the sequence. The subset with the highest *smoothing factor* becomes the set of outliers [5, 74]. The *cardinality* of each subset is used to scale the *smoothing factor*. The *cardinality* of a set is defined as the number of feature vectors in the set [5, 74]. The *smoothing factor* is calculated as [5]

$$sf(S_o) = |S_o - S_{o-1}| \times (D(S_o) - D(S_{o-1})) \tag{2.67}$$

  where $|\bullet|$ is the cardinality of the set, D is the function of dissimilarity, and the exception set S_e is defined as that set where [5]

$$sf(S_e) \geq sf(S_o) \quad \forall S_o \subset S \tag{2.68}$$

  Thus, the *smoothing factor* (sf), calculates the reduction in *dissimilarity* when removing a subset S_o of feature vectors from set S. If all feature vectors in S are *similar*, the *smoothing factor* is zero [5]. The exception set S_e has the highest sf value [5].

2.7 Alternative Computational Models for Clustering

This section discusses two alternative computational algorithms which can be applied to the problem of data clustering. Both of these algorithms implement neighbourhood topologies to influence the search behaviour of the algorithm. The model proposed in this thesis also utilises the concept of a neighbourhood topology. The two algorithms which are briefly discussed are the Particle Swarm Optimisation (PSO) algorithm, introduced by Kennedy and Eberhart [108, 109]

and the Self-organising Feature Map (SOFM or SOM), introduced by Kohonen [118].

2.7.1 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) algorithms model the formation and social behaviour found in bird flocks [108, 109]. PSO is a population-based stochastic search algorithm [109]. Thus, PSO maintains a population or a swarm of *particles*. Each *particle* represents a potential solution to an optimisation problem. In PSO, particles 'fly' through a multi-dimensional space in search of the *optimal* or *best* solution. The *best* solution to an optimisation problem is the particle with the highest *fitness*. The *fitness* of a particle is usually a function of the *objective* that needs to be optimised.

The position of each particle is presented by a feature vector in a multi-dimensional space. A particle moves through the search space by adjusting its position towards its own *best* experienced solution and towards the *best* particle in the *neighbourhood*. Since a particle needs to be able to adjust towards its own *best* experienced solution, a particle needs to maintain its *personal best position*. The *neighbourhood* can either be the entire swarm of particles or a subset thereof. The former case is known as *gbest* PSO and the latter as *lbest* PSO. In addition to the feature vector and *personal best position* contained by a particle, a particle also maintains its *current velocity*.

The rest of this section uses the following notation:

- S^N: The swarm or population of particles in N-dimensional search space;
- $\mathbf{x}_i$: The current feature vector or position of the i-th particle in S^N;
- $\mathbf{b}_i$: The i-th particle's *personal best position*;
- $\mathbf{v}_i$: The current velocity of the i-th particle in S^N;
- $\mathbf{V}_{max}$: The maximum allowed velocity of any particle in S^N;
- $\mathbf{g}_i$: The position of the *best* particle in the *neighbourhood* of the i-th particle in S^N;
- f: The fitness function (objective that needs to be optimised).

The i-th particle's position is adjusted by using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2.69}$$

where $\mathbf{v}_i(t+1)$ is the updated velocity of the particle at time step $t+1$. The velocity of the i-th particle is updated by using

$$\mathbf{v}_{i,n}(t+1) = w\mathbf{v}_{i,n}(t) + c_1 r_{1,n}(t)(\mathbf{b}_{i,n}(t) - \mathbf{x}_{i,n}(t)) + c_2 r_{2,n}(t)(\mathbf{g}_{i,n}(t) - \mathbf{x}_{i,n}(t)) \tag{2.70}$$

where w is the *inertia weight*, c_1 and c_2 are the acceleration constants, $r_{1,n}(t)$, $r_{2,n}(t) \sim U(0,1)$ and $n = 1,\ldots,N$. The velocity update in equation (2.70) basically consists of three components. These are:

- *inertia*: With the *inertia weight*, w, a fraction of the particle's previous velocity contributes to the update [160]. Large values of w result in better exploration of the search space, whereas lower w values result in better exploitation of the search space [160].
- *social component*: This is the $(\mathbf{g}_{i,n}(t) - \mathbf{x}_{i,n}(t))$ term, which is the *distance* in the n-th dimension to the *best* particle in a *neighbourhood*. The *best* particle $\mathbf{g}_i$ at time t, in a *neighbourhood* with radius d, is determined by using the following equation:

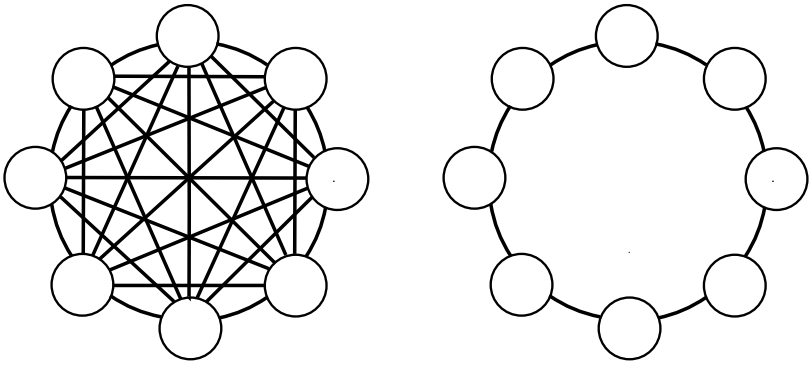
$$f(\mathbf{g}_i(t)) = min\{f(\mathbf{b}_{i-d}(t)), f(\mathbf{b}_{i-d+1}(t)), \ldots, f(\mathbf{b}_i(t)), \ldots, f(\mathbf{b}_{i+d}(t))\} \tag{2.71}$$

  If $d = \frac{|S|}{2}$, then the *neighbourhood* is the entire swarm of particles and the *best* particle at time t in the *neighbourhood* will be the same for all particles in the swarm. The resulting PSO is referred to as the *gbest* PSO. If $d < \frac{|S|}{2}$, then the *neighbourhood* is a subset of the swarm. The resulting PSO is referred to as the *lbest* PSO [160].
- *cognitive component*: This is the $(\mathbf{b}_{i,n}(t) - \mathbf{x}_{i,n}(t))$ term, which is the *distance* in the n-th dimension to the *personal best position* of the i-th particle. The *personal best position* of the i-th particle at time t is updated by

$$\mathbf{b}_i(t+1) = \begin{cases} \mathbf{b}_i(t) & \text{if} \quad f(\mathbf{x}_i(t+1)) \geq f(\mathbf{b}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if} \quad f(\mathbf{x}_i(t+1)) < f(\mathbf{b}_i(t)) \end{cases} \tag{2.72}$$

In order to limit the step size with which a particle's position is adjusted, the velocities can be clamped [45]. Therefore, if a particle's velocity exceeds the specified maximum velocity, $\mathbf{V}_{max}$, the particle's velocity is set to $\mathbf{V}_{max}$. The velocity of a particle, prior to the position update, is adjusted using,

$$\mathbf{v}_{i,n}(t+1) = \begin{cases} \mathbf{v}^*_{i,n}(t+1) & \text{if} \quad \mathbf{v}^*_{i,n}(t+1) < \mathbf{V}_{max,n} \\ \mathbf{V}_{max,n} & \text{if} \quad \mathbf{v}^*_{i,n}(t+1) \geq \mathbf{V}_{max,n} \end{cases} \tag{2.73}$$



(a) Star Topology (b) Ring Topology

Figure 2.4 Neighbourhood Topologies in PSO

where $\mathbf{v}^*_{i,n}(t+1)$ is calculated using equation (2.70) and $\mathbf{V}_{max,n}$ is the maximum allowed velocity in dimension n, controlling the granularity of the search. The values of $\mathbf{V}_{max}$ are selected as a fraction of the domain of each dimension of the search space, using

$$\mathbf{V}_{max,n} = \delta(\mathbf{x}_{max,n} - \mathbf{x}_{min,n}) \tag{2.74}$$

where $\mathbf{x}_{max,n}$ and $\mathbf{x}_{min,n}$ are the maximum and minimum values of the n-th dimension and $\delta \in (0,1]$. Figure 2.4 illustrates the two most common neighbourhood topologies used in PSO. These are the *star* and *ring* topologies [47]. The *star* neighbourhood topology is a fully meshed network of particles where every particle is connected to every other particle in the network topology. Each particle can therefore communicate with every other particle. The *ring* topology arranges particles in a ring structure such that each particle has a number of particles to the right and left forming the particle's neighbourhood.

PSO algorithms can be applied to the problem of data clustering [174, 145]. Algorithm 2.4 lists the pseudo code for a basic PSO clustering algorithm where t_{max} is the maximum number of iterations. Each particle in the swarm represents a possible partitioning of the data set. Thus, each particle represents K number of centroids, such that $N = K \times I$ where I is the number of features. Each particle is defined as: $\mathbf{x}_i = (\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \ldots, \mathbf{c}_{i,K})$ where $\mathbf{c}_{i,k}$ is the cluster centroid of the k-th cluster, $C_{i,k}$, represented by the i-th particle. In the context of clustering, the objective function optimised by the PSO, is defined as the quantization error [174]

$$J_{PSO} = \frac{\sum_{k=1}^{K} \frac{1}{|C_{i,k}|} \sum_{\forall \mathbf{p} \in C_{i,k}} \sigma(\mathbf{p}, \mathbf{c}_{i,k})}{K} \tag{2.75}$$

Algorithm 2.4: PSO Clustering Algorithm

Initialise each particle to contain K randomly initialised centroids;
for *t=1 to* t_{max} **do**
  for *each particle i* **do**
    for *each pattern* **p** **do**
      Calculate the Euclidean distance $\sigma(\mathbf{p}, \mathbf{c}_{i,k})$ (as defined in equation (2.3)) for all clusters $C_{i,k}$;
      Group pattern **p** with cluster $C_{i,k}$ such that $\sigma(\mathbf{p}, \mathbf{c}_{i,k}) = min_{k=1,\ldots,K}\{\sigma(\mathbf{p}, \mathbf{c}_{i,k})\}$;
    end
    Calculate the fitness of particle i;
  end
  Determine the *best* particle position, $\mathbf{g}_i$, in the swarm using equation (2.71);
  Determine the *personal best position*, $\mathbf{b}_i$, using equation (2.72);
  Update the cluster centroids of each particle using equations (2.69) and (2.70);
end

where σ is the Euclidean distance as defined in equation (2.3) and $|C_{i,k}|$ is the number of patterns grouped with cluster $C_{i,k}$. Thus, J is the fitness function of the particles, which is measured as the quantization error ($f = J_{PSO}$).

The same PSO algorithm in [174] was applied to image segmentation in [145], but with a different fitness function. An advantage of the PSO clustering algorithm compared to K-means clustering, is that the algorithm is less sensitive to the initialisation of cluster centroids, since PSO performs a parallel search for an optimal partitioning of the data set [145, 174]. The interested reader is referred to [47] for more information on swarm intelligence algorithms.

2.7.2 Self-organising Feature Map

The Self-organising Feature Map (SOM) is a single-layered unsupervised artificial neural network algorithm which consists of a single output layer known as a *map*, **M** [118]. The structure of the map is a two-dimensional *grid* of artificial neurons with R rows and C columns, $\mathbf{M}^{R\times C}$. The map is usually a square with $R = C$ but can also be any rectangular shape with $R \neq C$.

Each pattern, **p**, in a data set, P^N (where N is the number of dimensions), is associated with a single neuron in the map [96]. The number of neurons in the map is less than the number of patterns in the data set, i.e. $R \times C < |P^N|$. Each neuron in the map represents an N-dimensional

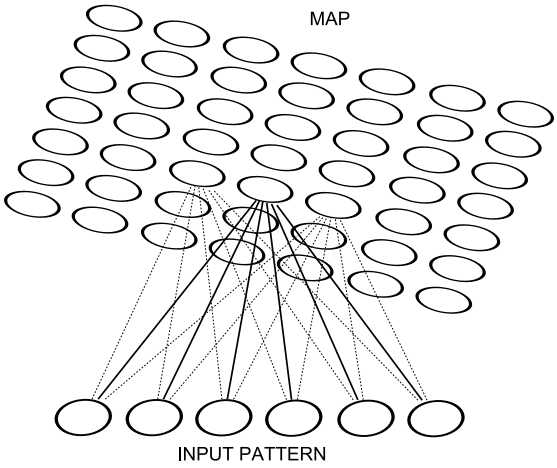


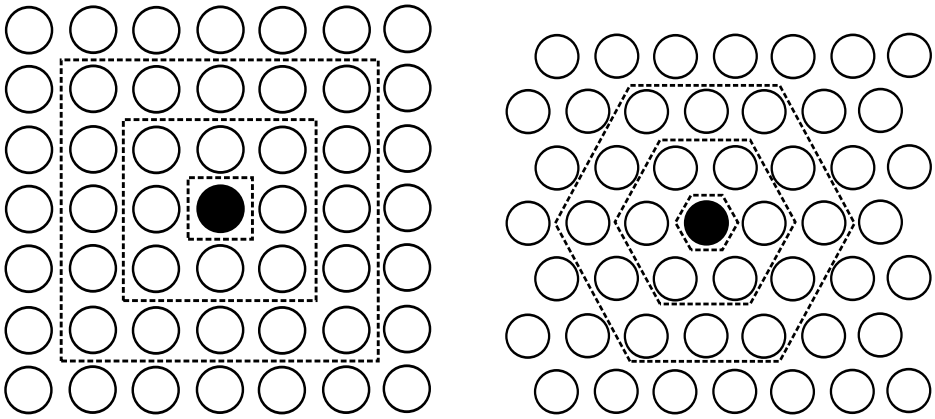
Figure 2.5 Self-organising Feature Map

weight vector, $\mathbf{w}_{rc}$, known as a *codebook* vector where r and c are the row and column indices, respectively. Figure 2.5 illustrates the association of an input pattern to the map of neurons. Training of the SOM is based on a *competitive* learning strategy where neurons compete to be the best matching neuron (BMN) to the input patterns which results in similar patterns being grouped together and represented by a single neuron [96]. Thus each codebook vector, $\mathbf{w}_{rc}$, forms the centroid of a cluster [48].

Algorithm 2.5: General SOM Algorithm

Initialise the learning rate $\gamma(0)$ and neighbourhood $\Lambda\left(\mathbf{w}',0\right)$;
Initialise the codebook vector of each neuron in the map, i.e. $\mathbf{w}_{rc}\ \forall r,c$;
repeat
  for *each input pattern* **p do**
    Determine the best matching neuron (BMN), $\mathbf{w}'$, using equation (2.3);
    Determine the neighbourhood, $\Lambda\left(\mathbf{w}',t\right)$, of BMN $\mathbf{w}'$;
    Using competitive learning, update the codebook vector of each neuron in neighbourhood $\Lambda\left(\mathbf{w}',t\right)$ by using equation (2.76);
  end
  Monotonically decrease the learning rate $\gamma(t)$;
  Reduce the neighbourhood $\Lambda\left(\mathbf{w}',t\right)$;
until *some stopping criterion is satisfied*;

Algorithm 2.5 lists general pseudo code of SOM training algorithms. There are various methods to initialise the codebook vector of each neuron. The two most common initialisation methods



(a) Square (b) Hexagon

Figure 2.6 Neighbourhood Arrangements in SOM

are:

- randomly initialise each codebook vector, or
- randomly select input patterns as initial codebook vectors.

The codebook vector of each *neighbouring* neuron is updated after each pattern is presented to the SOM. The neuron most *similar* to an input pattern is selected as the pattern's *best matching neuron* (BMN). Similarity between an input pattern $\mathbf{p}$ and a neuron's codebook vector $\mathbf{w}_{rc}$, is measured with the Euclidean distance as defined in equation (2.3). Thus the BMN, $\mathbf{w}'$, of an input pattern, $\mathbf{p}$, is the minimum Euclidean distance to the input pattern. The BMN and its neighbouring neurons are then moved closer to the input pattern by updating the codebook vectors. The neighbourhood of a BMN can be arranged as a square or hexagon lattice (as illustrated in figure 2.6) or can be determined by using a smooth Gaussian function [48]. The codebook vector, $\mathbf{w}_{rc}$, of a neighbouring neuron is updated by using

$$\mathbf{w}_{rc}(t+1) = \begin{cases} \mathbf{w}_{rc}(t) + \gamma(t)\left[\mathbf{p} - \mathbf{w}_{rc}(t)\right] & \text{if} \quad \mathbf{w}_{rc}(t) \in \Lambda\left(\mathbf{w}', t\right) \\ \mathbf{w}_{rc}(t) & \text{otherwise} \end{cases} \tag{2.76}$$

where $\gamma(t)$ is the learning rate and $\Lambda\left(\mathbf{w}', t\right)$ is the set of neighbourhood neurons of the best matching neuron, $\mathbf{w}'$, for input pattern $\mathbf{p}$ at time step t. The neighbourhood radius decreases with each time step to reduce the influence of distant neighbouring neurons. The iterative learning process stops when one of the following criteria is met:

- the neighbourhood $\Lambda\left(\mathbf{w}', t\right)$ only includes the BMN $\mathbf{w}'$,

- the maximum number of time steps (iterations) is exceeded,
- there are no changes in the codebook vectors, or
- the quantization error is sufficiently small. The quantization error is calculated as the sum of Euclidean distances between all input patterns and the codebook vector of BMN, defined as [48]

$$J_{SOM} = \sum_{\forall \mathbf{p} \in P} \sigma\left(\mathbf{p}, \mathbf{w}'(t)\right) \tag{2.77}$$

A SOM maps multi-dimensional data onto a two-dimensional map which is a much simpler representation of the data and is easier to interpret. Thus, a SOM maintains the topology of the data [48]. There are however a few drawbacks to SOMs [48, 96]:

- The random initialisation of codebook vectors can result in increased training times.
- Initialisation of codebook vectors to randomly selected input patterns may result in premature convergence.
- SOMs are only applicable to clustering hyper-spherical data.
- The clustering result is dependent on the number of neurons in the map.

In order to determine the clusters within the data set, the boundaries between clusters in the map need to be identified. Boundaries in the map can be determined by means of the unified distance matrix (U-matrix) technique or Ward clustering technique [3]. The former is a matrix which contains a geometrical approximation of the codebook vector distribution in the map. The latter technique is an agglomerative hierarchical clustering method which partitions the codebook vectors into a specified number of clusters. The linking of two adjacent clusters is based on the Ward distance measure which is calculated as

$$\ell_{Ward}\left(C_i, C_j\right) = \frac{|C_i| \times |C_j|}{|C_i| + |C_j|} \sigma\left(\mathbf{c}_i, \mathbf{c}_j\right) \tag{2.78}$$

where σ is the Euclidean distance measure as defined in equation (2.3). The two clusters with the smallest ℓ_{Ward} distance are merged and the centroid of the new cluster, $C_k = C_i \cup C_j$, is calculated as

$$\mathbf{c}_k = \frac{1}{|C_i| + |C_j|} \left(|C_i| \mathbf{c}_i + |C_j| \mathbf{c}_j \right) \tag{2.79}$$

2.8 Conclusion

The chapter gave a formal definition of data clustering and discussed different distance-based similarity measures which can be used to determine the similarity between two feature vectors. The chapter also discussed the most familiar clustering algorithms which are categorised into hierarchical or partitional clustering methods. This was followed by an overview of different cluster validity indices which evaluate the partitioning quality of a clustering algorithm.

Since the cluster quality of a clustering algorithm can be influenced by outliers in a data set, a brief introduction was also given on outlier mining and different techniques for outlier detection were discussed. Another influence on the cluster quality of a clustering algorithm is a changing environment. This means that the feature vectors are non-stationary. Thus, the cluster quality of a clustering algorithm needs to be measured over a period of time. Therefore, the chapter discussed existing measures to determine the performance of optimisation algorithms in a non-stationary environment. These performance measures are used to quantify and define performance measures that can be used to evaluate the cluster quality of a clustering algorithm in non-stationary environments.

The chapter ended with a discussion of the Particle Swarm Optimisation and Self-organising Feature Map algorithms and how these algorithms can be applied for clustering data. Both of these algorithms implement different neighbourhood techniques to adapt solutions to feature vectors in the data.

The proposed model in this thesis is inspired by and based on the network theory in immunology. The proposed model is a partitional clustering method which also implements a neighbourhood technique and uses the Euclidean distance as similarity measure between two feature vectors (discussed in chapter 5). The cluster validity indices of Davies-Bouldin and Ray-Turi, which are defined in equations (2.41) and (2.51) respectively, are used to evaluate the clustering quality of a clustering algorithm in this thesis. These cluster validity indices were selected since an optimal partitioning of a data set minimises all of these indices. An enhancement to the proposed algorithm in this thesis implements the sequential exception technique as discussed in section 2.6 to determine the boundaries between clusters and thereby dynamically determines the number of clusters within a data set (discussed in chapter 6). Both the original and enhanced version of the proposed clustering algorithm are also applied to clustering of non-stationary environments (discussed in chapter 7). Section 7.1 revisits the clustering performance measures discussed in

this chapter and proposes a clustering performance measure for non-stationary data clustering.

Since the proposed model in this thesis is inspired by the network theory in immunology, the next chapter reviews the functional process of the natural immune system and discusses the different theories in immunology regarding the functioning and organisational behavior between lymphocytes.