Learning feature weights for K-Means clustering using the Minkowski metric

Renato Cordeiro de Amorim
Department of Computer Science and Information Systems

Birkbeck, University of London

April 2011

Thesis submitted in fulfilment of requirements for degree of PhD

Declaration

I hereby declare that the work presented in this thesis is my own, and that it has not previously been submitted for a degree or award at this or any other academic institution.

Signed:

Renato Cordeiro de Amorim

Abstract

K-Means is arguably the most popular clustering algorithm; this is why it is of great interest to tackle its shortcomings. The drawback in the heart of this project is that this algorithm gives the same level of relevance to all the features in a dataset. This can have disastrous consequences when the features are taken from a database just because they are available. Another issue of our concern is that K-Means results are highly dependent on the initial centroids.

To address the issue of unequal relevance of the features we use a three-stage extension of the generic K-Means in which a third step is added to the usual two steps in a K-Means iteration: feature weighting update. We extend the generic K-Means to what we refer to as Minkowski Weighted K-Means method. Specifically, we use feature weights as scaling coefficients under Minkowski Lp metric at $p \geq 1$ and, thus, attend to a three-stage method for alternating minimisation of Minkowski K-Means criterion over three groups of variables: (a) feature weights, (b) centroids, and (c) cluster lists. We extend Mirkin (2005) method of anomalous clusters to initialize the centroids and feature weights in this setting. In our experiments on both synthetic and real-world data, MWK-Means method proves superior over those previous ones on several counts, the most important being the drastic reduction of impact by noisy features.

To address the issue of feature selection in this context, we develop a method, iKFS, an extension of a popular method by Mitra et al. (2002), in which we take into account the cluster structure of the feature set. Another advantage of iKFS is that it involves no user-specified parameters.

We apply the developed approaches to problems in distinguishing between different mental tasks over high-dimensional EEG data.

Contents

Acknowledgements

My most sincere thanks go to my supervisor Prof. Boris Mirkin. His constructive criticism and profound knowledge regarding clustering (and history!) have surely made our meetings rather interesting.

Thanks are also due to Prof. John Q. Gan for multiple consultations regarding EEG data and trial classification, as well as for supplying the datasets we use in the experiments in Chapter 5.

There are surely a number of other people who I would also like to thank: Prof. Trevor Fenner and Prof. George Magoulas for their insightful comments regarding the progress of this research, Phil Greg for making whatever he could to make sure I always had enough computing power and Sam Szanto for the grammatical review of this thesis.

Last but not least, this thesis would have never been possible without the strong support of my parents and grandparents, to who the deepness of my gratitude cannot be expressed in words.

List of tables

Table B.7: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise dataset (ix) - 1000x50, including 50 extra noise features.

Table B.8: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise, dataset (x) - 1000x75, including 25 extra noise features.

List of figures

List of publications

Amorim, R. C. (2008), Constrained Intelligent K-Means: Improving Results with Limited Previous Knowledge, *Proceeding of Advanced Engineering Computing and Applications in Sciences*, IEEE Computer Society, pp 176-180.

Amorim, R. C., Mirkin, B., Gan J. Q. (2010), A Method for Classifying Mental Tasks in the Space of EEG Transforms. *Technical report BBKCS-10-01*, Birkbeck University of London, London.

Amorim, R. C., and Mirkin, B. (2011), Minkowski Metric, Feature Weighting and Anomalous Cluster Initializing in K-Means Clustering. *Pattern Recognition* (in revision).

List of equations

Chapter 1: Introduction

1.1 Introduction and objectives

Advances in technology have allowed the storage of massive amounts of data. Organisations and even ordinary people can easily acquire and keep larger quantities of data than ever before. This is evident if one compares the capacity of hard disks sold in common computers ten years with the ones we have today.

Data on its own is unlikely to be useful; the user need is normally towards information. For large datasets we have the whole field of data mining, which takes into account the intuitive idea that the larger the amount of data, the more difficult it is to extract information from it. A number of approaches can be used in data mining, and clustering stands as a popular one. Clustering separates data without the need of labelled samples; this particularly distinguishable characteristic associates it with unsupervised learning.

It is fair to assume that in a dataset not all features are necessarily relevant or correct, by consequence clustering may benefit from using only a selected subset of the features. This point has been taken further in a number of papers, including DeSarbo et al. (1984), Green et al. (1990), Strehl et al. (2000), Makarenkov and Legendre (2001), Modha and Spangler (2003), Xing et al. (2003), Chan et al. (2004), Frigui and Nasraoui (2004), Huang et al. (2005, 2008), and Tsai and Chiu (2008). They assume that not all features, even if they are relevant, will necessarily have the same degree of relevance. Instead of selecting features using a binary approach, they generalise this idea and set weights to the features in the range [0, 1]. As expected, setting feature weights without labelled samples for an algorithm to learn from is not a trivial task. This is probably the reason why the above referenced papers use at times very different approaches, as we

discuss in Chapter 3, Section 3.2.

This thesis aims to further the research in feature weighting, specifically analysing the cases of noise features and similar features, allowing the use of the Minkowski metric (formally introduced in Chapter 2, Section 2.2). Generally speaking, this metric allows identifying, and in fact, ignoring irrelevant features (Kivinen et al., 2006). We intend to:

(i) Answer the question:

Can the use of the Minkowski metric improve the accuracy of a weighted clustering algorithm?

Specifically, we:

a. Review generic clustering, and more specifically feature weighting clustering algorithms, selecting one of the latter.
b. Introduce an extension to the chosen feature weighting clustering algorithm that will allow it to use the Minkowski metric.
c. Introduce a deterministic version of the chosen feature weighting algorithm, yet using the Minkowski metric
d. Define an evaluation criterion, aligned to what has been used in the literature of the chosen feature weighting clustering algorithm.
e. Acquire and generate datasets, with and without noise features.
f. Compare the introduced algorithms to well established counterparts
g. Investigate the possibility of learning the Minkowski exponent from a small amount of labelled data.

(ii) We analyse the impact of similar features in clustering weighted methods. To do this we:

a. Apply the weighted algorithms to datasets in a high-dimensional space. Such datasets are more likely to have similar features than those in a low-dimensional space.
b. Apply a popular method for similar feature reduction, and re-run the experiments.
c. Improve the chosen feature reduction algorithm to remove the necessity of any extra parameter, and make sure it takes the feature structure into consideration.
d. Compare the new feature reduction algorithm with the chosen popular one.

1.2 Contribution

In order to meet the objectives, this thesis introduces two clustering methods: Minkowski Weighted K-Means and intelligent Minkowski Weighted K-Means. These are formally introduced in Chapter 3, Sections 3.3.2 and 3.3.3 respectively. They automatically assign *K* weights to each feature in a dataset, where *K* is a pre-defined number of clusters.

We also develop a new method for reducing the amount of features based on their similarity, the intelligent K-Means with feature similarity (iKFS), formally introduced in Chapter 5, Section 5.4.2. In our method we assume that features clustered together carry similar information and by consequence most can be discarded. If used as a pre-step, it potentially increases the accuracy of our algorithms in high-

dimensional spaces as Chapter 5, Section 5.5.2 shows.

We conduct extensive experiments to substantiate the use of the introduced methods.

1.3 Thesis structure

Chapter 2 presents a literature review of clustering as a whole, including algorithms that are arguably the most important in the field. In this chapter we justify our choice of K-Means and iK-Means.

Chapter 3 discusses our new methods for feature weighting using the Minkowski metric. It aims to describe two methods and show their superiority through a comparison with other well established methods.

Chapter 4 deals with the open issue of finding an optimal Minkowski exponent for a given dataset. In this we propose a method bases on semi-supervised learning.

Chapter 5 relates to the application of our method in a high-dimensional space. We also experiment with a popular algorithm used to discard similar features. We introduce a method with a similar aim that unlike others does not have an extra parameter.

Chapter 6 summarizes the contribution of this thesis, discusses its limitations, and sets possible paths for future research.

Chapter 2: K-Means based clustering, a review

2.1 Introduction

In his book, Hartigan (1975:1) defines clustering as the grouping of similar objects. The meaning of this brief, but precise, definition may not be clear at first. For instance, it merely touches upon the loose notion of similarity, that in order for us to group together similar objects we firstly need a method to quantify the similarity of two objects. In clustering, this is normally done with a distance (also called dissimilarity) measure. Since this measure will be used on the objects' features, it can also be concluded that clustering will depend on such features, and is therefore a data-driven task. As stated by Jain and Dubes (1988:1), the goal of clustering is to produce groups containing objects that are alike, while objects in different groups are not like. The clustering task attempts to empirically elucidate the 'natural', unknown and ideally interesting groups within data, making their structure implicit.

Humans quite often perform clustering: most of the time without noticing. For instance, sorting documents into folders normally involves grouping similar ones together. Clustering is also a common task in science, which has been applied to a number of different fields, such as: data mining, computer vision, bioinformatics (Brohee and Helden, 2006), crime analysis (Hartigan, 1975:30), and the visualisation of data streams (Wong et al., 2004). Some scholars, including Romesburg (2004:4), take the view that clustering may be useful to researchers in all fields, as all will need to make and revise classifications.

The clustering task is usually performed under a condition that is known as unsupervised learning. This type of learning does not require object samples to be labelled in order to group other, similar objects – in contrast to supervised learning. Clustering algorithms *learn* from the objects to be grouped and their features, rather

than from the given samples. This can be seen as a considerable advantage, as although acquiring data is much easier nowadays than just a few decades ago, labelling such data can still be difficult and error prone. Also, algorithms based on supervised learning that receive wrongly labelled samples from which to learn, or simply receive an insufficient amount of correctly labelled samples, are likely to underperform.

Of course, under ideal conditions, algorithms based on supervised learning are likely to be closer to an optimal result that those based on unsupervised learning. We can intuitively recognise that it tends to be easier to learn from given samples than from no samples. The problem is that such ideal conditions cannot always be met. As has been shown by Duda et al. (2001:517), acquiring recorded speech can have a very small, at times even negligible, cost, but labelling such speech word by word or phoneme by phoneme can be very expensive and time consuming. In scenarios where obtaining a representative quantity of labelled data is unfeasible, or ones in which labels do exist but are unreliable, supervised learning becomes impractical.

Another interesting aspect of unsupervised learning is that it does not need to be the only learning approach. Different learning approaches should not be seen to be in competition. For instance, clustering can be combined with a supervised learning method (Callan, 2003:312) to generate a two-phased learning approach for classification or regression. We can, of course, expand this suggestion to semi-supervised learning, which is comprised of methods that normally attempt to learn from a very small amount of labelled data.

In his monograph, Mirkin (2005:3) presents a list of five objectives for clustering, acknowledging these may overlap to some degree: they are: (i) *structuring*: the objective of which is to find the primary groups of objects and represent the data as a set of these groups; (ii) *description*: describing the clusters in relation to their features

– within this objective, finding clusters is not a major concern; the desirability of this description stems from the fact that a good cluster description may be used to enhance understanding of a cluster, or even for better prediction; (iii) *association*: uncovering relationships between clusters and, by consequence, objects – clusters are said to be related if they are well described twice, but each description relates to a different aspect; (iv) *generalisation*: deriving general statements about data – as pointed out by Hartigan (1988:6), one may refer to the features of clusters rather than to the features of individual objects; (v) *visualisation*: a rather general term for mapping data structures onto a known ground image, in most cases a coordinate place. Jain (2010) states, rather more succinctly, that the overarching aim of clustering is to find structure in data and that, as a consequence, it is exploratory in nature.

After the clustering algorithm identifies the groups of data, which we call clusters, it should assign objects to them. Witten and Frank (2005:137) define four categories to this assignment. Each category has a direct impact on the final cluster structure:

- *Non-overlapping*

  Each object belongs to one cluster only.

- *Overlapping*

  Any object may belong to more than one cluster.

- *Probabilistic*

  An object belongs to each cluster with a certain probability.

- *Hierarchical*

  In this category, the cluster structure looks like a tree in which a cluster with all its objects represents the root. These clusters can be created

following either a top-down, or a bottom-up, approach.

Clearly, clustering is not without flaws, and the inexactitude begins with the fact that there is no universally agreed definition for the term *cluster* (Xu and Wunsche, 2010:4), so it becomes then subjective to the problem. Some scholars argue that this lack of a key definition is the main reason why there are so many clustering algorithms today (Estivill-Castro, 2002). Also, finding the number of clusters in a dataset can be quite problematic, as a wide variety of algorithms assume that such number is known beforehand. The difficulty in defining the term *cluster* stems from the fact that the actual number of clusters depends on a series of factors, such as:

- The resolution being used in the analysis (*fine versus coarse*) (Jain, 1988:2; Nascimento, 2005:2); a change in resolution may then generate a change in the number of clusters.
- The method being applied. Different algorithms will often result in different groupings (Mingoti and Lima, 2006) as each implicitly imposes a structure upon the data (Webb, 2002: 361).
- The distance (sometimes called the dissimilarity) measure, as in some cases different measures can be incorporated into the same method. The issue is that there are a number of different distance measures, and each may look for a different geometrical shape in the data (Pedrycz, 2005: 26). A consequence is that the clusters and their quantities may then differ.

Although the groups will not necessarily accurately represent the inner structure of the data, this is the main aim of clustering. This may happen because clustering does

not guarantee that the error will reach its global minima. Also, there may be scenarios in which the data do not present 'natural' grouping, but clustering algorithms will group objects anyway (Webb, 2002:361). Finally, after the clustering process is finished, it is difficult to determine its success, as in most real-world scenarios there are no labelled test data to compare results (Hand et al., 2010:295).

2.2 Clustering algorithms

Because of the rather large amount of research that has been conducted in the field, there is now a vast quantity of clustering algorithms. Comprehensive reviews of examples of such algorithms can be found in the works of: Hartigan (1975), Aldenderfer and Blashfield (1984), Jain and Dubes (1988), Kaufman and Rousseeuw (1990), Mirkin (1996, 2005), Romesburg (2004), Pedrycz (2005), Xu and Wunsch II (2005, 2010), Berkhin (2006), and Jain (2010).

As previously discussed, the results of clustering algorithms may depend on the distance measure in use, of which there are a large number. Deza and Deza (2009) have published a encyclopaedia on the subject, while Perlibakas (2004) presents a shorter, but still comprehensive review on the subject, focusing in the field of face recognition.

Probably the most common measure used in clustering algorithms is the squared Euclidean distance. Assuming two points, *x* and *y*, of N dimensions, this can be defined as follows:

$$d(x,y) = \sum_{i=1}^{N} (x_i - y_i)^2$$

Equation 2.1: Squared Euclidean distance

The Euclidean distance is rather intuitive as it can be viewed through the Pythagoras' Theorem.

The Hamming distance (also known as the city block distance and Manhattan distance) is in common use as well. In this, the distance is defined by the absolute difference between two points, as in the equation:

$$d(x, y) = \sum_{i=1}^{N} |x_i - y_i|$$

Equation 2.2: Hamming (also called city block and Manhattan) distance

The Minkowski distance is a generalisation of both the Euclidian and Hamming distances; it is as follows:

$$d(x, y) = \sqrt[p]{\sum_{i=1}^{N} (x_i - y_i)^p}, p > 0$$

Equation 2.3: Minkowski distance

If *p* is equal to one or two, the results are equivalent to the Hamming distance and Euclidian distance, respectively. Figure 2.1 shows the effect of different *p* on the distance.

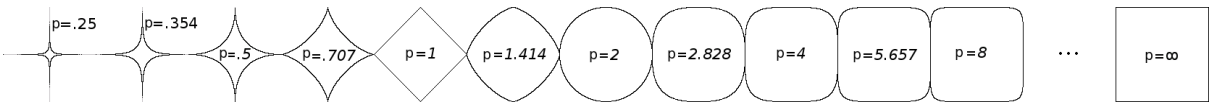

Figure 2.1: A visual representation of the Minkowski distance "circles", a set of points equidistant to the centre, at different *p*. This public domain image can be found in http://en.wikipedia.org/wiki/File:Minkowski2.png

Clustering algorithms are commonly divided into Hierarchical and Partitional clustering, and these are explained in sections 2.2.1 and 2.2.2 respectively.

2.2.1 Hierarchical clustering

Hierarchical clustering produces cluster structures which are often represented as trees. These structures, also known as dendograms, allow a root cluster to have branch clusters (each may later become a root cluster and generate more branches). The dataset features in this type of clustering do not need to be in the same hierarchy.

In hierarchical clustering, singletons (clusters containing a single object) are referred to as terminal nodes, and the universal combined cluster consisting of the entire entity set is referred to as the root of the tree; all other clusters are referred to as its nodes. Hierarchical clustering can be subdivided into:

(i) A bottom-up approach known as agglomerative clustering. When using the agglomerative method, the clustering task normally begins with a set of singletons. Algorithms in this category will recursively merge the two or more most similar clusters.

(ii) An inverse, top-to-bottom approach, known as divise clustering. Algorithms in this category normally initialise with the root cluster. They recursively split the most appropriate cluster into smaller units.

Algorithms in both the agglomerative and divisive categories will continue until a stopping criterion is met. Berkhin (2006) identifies the vagueness of the termination criteria, as well as the fact that most hierarchical algorithms do not revisit clusters once they are constructed, as the major disadvantages of hierarchical clustering. On the other hand, Berkhin also identifies advantages to this approach, such as the flexibility given

regarding the level of granularity, which can be tuned in the stopping criteria and the easy means of handling any form of similarity or distance measure. Hierarchical clustering includes algorithms such as single and full linkage, spanning tree, etc.

As hierarchical clustering will always merge or split clusters, the method used to determine the distance in between clusters (not to be confused with the previously discussed distance measure between points) is of key importance. The use of different methods is a major difference between algorithms as they may, just like distance measures, impact on an algorithm's final outcome. In other words, algorithms using different methods may produce different cluster structures. There are a number of such methods, including:

(i) Single linkage (also called nearest neighbour), which defines the distance between two clusters as the minimum possible. The method finds the two entities, one of each cluster, that are the closest to each other, as per the equation below:

$$D(X,Y) = \min_{x \in X, y \in Y} d(x,y)$$

Equation 2.4: Single linkage distance

in this equation, the distance between clusters X and Y is given by the distance in between the two closest entities, $x \in X$ and y $\in Y$, in the two clusters. A disadvantage of this rather simple method is that having two close entities does not necessarily mean that the whole two clusters can be considered to be close.

(ii) Full linkage, which uses the opposite of the above. The measure finds the two entities, $x \in X$ and y $\in Y$, that are the farthest from each other.

$$D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$$

Equation 2.5: Full linkage distance

In terms of disadvantage, we can refer back to the previous method and conclude that having two entities, $x \in X$ and y $\in Y$, which have the maximum distance between entities of clusters *X* and *Y*, small, does not necessarily mean that the two clusters are close.

(iii) Group average criterion (GAC) defines the distance between two clusters as the distance between their centroids. Assuming the centroids are the average of a cluster, one can define the GAC as follows:

$$D(X,Y) = \frac{1}{|X|.|Y|}\sum_{x\in X}\sum_{y\in Y} d(x,y)$$

Equation 2.6: Group average criterion

(iv) The Ward method (Ward, 1963) calculates the distance between two clusters, taking into account the number of elements in each.

This section will now exemplify hierarchical clustering algorithms based on the Ward distance, as it is the only one from the above with a non-obvious implementation. Given two clusters S_{w1} and S_{w2}, represented by the centroids c_{w1}, c_{w2}, and with a total number of elements N_{w1}, N_{w2}, respectively, the Ward distance would be defined as:

$$dw(S_{w1}, S_{w2}) = \frac{N_{w1}N_{w2}}{N_{w1} + N_{w2}} d(c_{w1}, c_{w2}),$$

Equation 2.7: Ward distance

where $d(c_{w1}, c_{w2})$ is normally the squared Euclidean distance between the two centroids. Mirkin (2005: 115) illustrates the agglomerative Ward algorithm as follows:

1. *Initial setting*
   Each entity is set as a singleton cluster with the centroids being themselves; the initial set of maximal clusters contains all of them.
2. *Cluster update*
   The two clusters with the smallest Ward distance within the maximal clusters are merged. The new parent cluster contains all entities of the two initial clusters, and its cardinality is equal to the sum of the previous two child cluster cardinalities. The new centroids are defined by: $c_{w1\cup w2} = (N_{w1}c_{w1} + N_{w2}c_{w2})/N_{w1\cup w2}$.
3. *Distance update*
   Remove the child clusters found in step 2 from the maximal clusters, and replace them with the new parent cluster. Calculate the distance between the new parent cluster and all other maximal clusters.
4. *Repeat*
   If there is more than one maximal cluster, return to step 2; otherwise, finish here.

Algorithm 2.1: Agglomerative Ward algorithm

In order to use single or full linkage, change steps 2 and 3, the *cluster update* and *distance update* to use equations 2.4 and 2.5 respectively. This way, the centroids would be the average of their cluster and the distance measure used would cease to be Ward's distance and be either the minimum or maximum distance.

The Ward criterion can also be used in divisive clustering. In this approach, the objective is to split a cluster in two, rather than to merge them. Mirkin (2005: 117) describes the algorithm as follows:

1. *Start*

   Put the universal cluster containing all entities into S_w.
2. *Splitting*

   Split cluster S_w into two sub-clusters, namely S_{w1} and S_{w2}, to maximise the Ward distance between them, as per equation 2.7.
3. *Drawing attributes*

   Add the two new child nodes, S_{w1} and S_{w2}, to S_w.
4. *Cluster set's update*

   Get a new $S_w$by setting it equal to the node of the maximum height of the current upper cluster hierarchy.
5. *Stop*

   If the stop condition does not hold, return to step 2. Arguably, the most common stop criterion is the number of leaves reaching a threshold specified by the user.

Algorithm 2.2: Divisive Ward algorithm

2.2.2 Partitional clustering

Partitional clustering includes algorithms with objective functions to be optimised, normally in an iterative way. This is done because, in most scenarios, checking all possible cluster structures is computationally infeasible. Unfortunately, such an approach tends to lead to a suboptimal configuration of disjointed clusters, which is nonetheless appealing in practice (Pedrycz, 2005). Each found cluster is nontrivial, for instance containing some objects but not all; at the same time, the clustering structure should not be a set of clusters containing a single object, otherwise known as singletons. Berkhin (2006) identifies as a major advantage of partitional clustering, the fact that iterative optimisation may gradually improve clusters; he states that with appropriate data, this results in high quality clusters. This is unlike hierarchical clustering, as algorithms under that class do not feature re-visits to clusters.

There are a number of well-known algorithms in partitional clustering, including those under the probabilistic approach (EM framework), and others such as K-Medoids (PAM, CLARA, etc) and the original K-means and its many variations regarding, for instance, initialisation, feature selection and optimisation.

K-Means (also known as moving centres or straight K-means) originated independently in the works of MacQueen (1967) and Ball and Hall (1967). K-Means is arguably the most popular clustering algorithm, which produces non-overlapping clusters. It is believed to be more efficient than the hierarchical algorithms (Manning et al., 2008:335). Each cluster has a centroid (also known as a prototype or seed), which represents the general features of the cluster.

Give a number *K* of clusters of a dataset containing a set of *N* entities, *I*, and *M* measurements, *V*, and a quantitative entity-to-feature, *Y*=(y_{iv}), such that y_{iv} represents the value of feature $v \in V$ at entity $i \in I$. This algorithm attempts to optimise the clusters by minimising the sum of the within-cluster distances of the *N* entities to their respective centroid $c_k = (c_{kv})$ where *k=1, 2, ... K*: it produces a partition, S=$\{S_1, S_2, \ldots, S_K\}$. Each partition is non-overlapping, so each entity *i* is assigned to only one partition, a non-empty subset of S – a cluster.

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(i, c_k)$$

Equation 2.8: General K-means criterion

The distance measure represented by *d* in Equation 2.8 is usually the squared Euclidian distance, which can be re-written as follows:

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} s_{ik}(y_{iv} - c_{kv})^2$$

Equation 2.9: General K-means criterion using the squared Euclidian distance

in the above, s_{ik} is a Boolean variable representing the cluster membership of i to the cluster k. Formally, s_{ik} will be equal to one iff $i \in S_k$. In clustering, any chosen score function will impact on the homogeneity of the clusters. K-means is guaranteed to converge, as demonstrated by Manning et al. (2008:334), but the minimisation of its score function is known to be NP-hard (Drineas, 1999).

The score function of K-means is clear, but unfortunately there is no closed-form solution, which means that the error of the final outcome is not necessarily a global minimum. In real-world problems, it is unlikely to be possible to complete an exhaustive search for all clustering scenarios, so K-means then becomes essentially a search problem (Hand et al., 2001:302). In such a scenario, it is feasible to use an iterative-improvement algorithm, and K-means bases this in expectation maximisation (EM). Mirkin (2005:80) defines K-means in four steps:

0. *Data pre-processing*

   One of the main weaknesses of K-means is its inability to deal with non-numerical attributes (Berkhin, 2006; Xu and Wunsch, 2005). Consequently, all data should be transformed into a quantitative matrix, by converting qualitative categories, standardising the data, and rescaling (Mirkin, 2005:64). This step should not be overlooked, as pre-processing, as well as post-processing can be as important as the clustering algorithm itself (Xu and Wunsch, 2005).

1. *Initial setting*

   Define the value of *K* and the tentative centroids $c_1, c_2, \dots, c_K$. In the conventional K-means, the initial centroids are usually randomly chosen entities. As per the value of *K*, this is a difficult question. Its value may come from any expert knowledge relating to the data or from experiments at different values.

2. *Cluster update*

   Assign the *N* entities to their respective centroid, $c_k = (c_{kv})$, using the minimum distance rule.

3. *Stop condition*

   The most common stop condition is to check for changes to the clusters in step 2: should there be none, the clustering task is assumed to be finished and the generated partitions S=$\{S_1, S_2, \dots, S_K\}$ are final. Other possible stop conditions are limiting the number of iterations or pre-setting a threshold for the objective function.

4. *Centroids update*

   For each cluster, move its centroid to its cluster's centre of gravity. This way, each will represent the general properties of its cluster. When using the squared Euclidian distance, this can be achieved by simply moving the centroids to the means of their clusters.

Algorithm 2.3: K-Means

K-means has been used to solve numerous problems since the 1960s. After such amount of time and work, it is no surprise that its weaknesses are well known. In terms of speed, although the required number of iterations tends to be less than the number of

entities (Duda et al., 2001:527), Arthur and Vassilvitskii (2006) demonstrate that in the worst-case scenario, K-mean's running time can be superpolynomial.

The algorithm is also highly dependent on the *K* initial centroids. These are obtained randomly and it is, therefore, possible to choose bad initial centroids, such as outliers that are likely to generate singletons, or even a set of initial centroids that are simply too close to each other. Also, finding the actual value of *K*, or the cardinality of a clustering, may be problematic. The most common solution to both problems is to run the algorithm a pre-specified number of times at different *K* and to perform an analysis of the results.

Another issue is the fact that the algorithm considers all measurements at the same level, disregarding the fact that, for instance, the data may be contaminated. Mingoti and Lima (2006) empirically demonstrate the effect of such contamination in K-Means. There may also, more simply, be a large number of unnecessary features of the data.

The above weaknesses of K-means will be further discussed in the next chapter of this thesis; unfortunately, these are not the only weaknesses. As a greedy algorithm, K-means is not always expected to converge to a global minimum, as previously discussed here, and by Berkhin (2006) and Xu and Wunsch II (2005). Hartigan and Wong (1979) present a classical solution to this problem by swapping entities between clusters.

Despite its many critical weaknesses, this algorithm also has advantages. K-means may produce tighter clusters than a hierarchical counterpart. This is especially true when the cluster shapes are well aligned with the distance function: for instance, spherical shapes when using squared Euclidian distance. Arguably, the main reasons

for its popularity are related to its ease of implementation, simplicity, efficiency, and empirical success (Jain, 2010).

The partition around medoids (PAM) algorithm is rather similar to K-means. They are both partitional clustering algorithms that attempt to minimise the within-cluster distances of entities to the cluster's centroid (Equation 2.8). Neither guarantee an optimal solution and by consequence may arrive instead at a local minimum. Also, they both assume that the number *K* of clusters is known beforehand. The difference between them lies in the definition of a prototype. While in K-means, the prototype is a centroid in the centre of gravity of a cluster, normally its average, in PAM each cluster has a medoid as a prototype; these are entities initially chosen at random that have the smallest sum of distances to all other entities within the same cluster. The only change in the K-means algorithm would be in step four, which then becomes:

> *4. Medoids update*
> For each cluster, choose as a medoid the entity that has the minimal average distance to the other entities within the same cluster. This way, each will represent the general properties of its cluster.

Algorithm 2.4: Partition around medoids (PAM). One-step update of the original K-means

There are advantages to using actual entities (medoids) as prototypes, instead of the centre of gravity of clusters, normally the cluster average (centroids): for instance, the latter may be too artificial for certain scenarios (Mirkin, 2005:181). Also, by using real entities, PAM may be able to avoid the effect of outliers to the resulting prototypes (Xu and Wunsch II, 2010:72).

The K-Median clustering algorithm is another common variation of K-Means. The difference is again in the way the prototype is calculated; in K-Median, as the

name suggests, the prototype is the median rather than the centre of gravity. In this algorithm, the outliers also have a smaller influence on the centroids, and this may be the reason why K-Median is regarded by some as comparable to, or better than, K-Means (Bradley et al., 1997, and references within).

Here, the specifications of the PAM and K-Median algorithms are purposely similar to K-Means, to make it easier to visualise any change made to the K-Means algorithm being easily adaptable to them.

2.3 Use of K-Means

Clustering has been addressed in research since the 1960s. Some of the first related work took place by scholars in the fields of factor analysis (Holzinger, 1941), numerical taxonomy (Sneath and Sokal, 1973), and unsupervised learning in pattern recognition (Duda and Hart, 1973).

Nowadays, clustering is used in numerous fields. In bioinformatics, it is used in microarray data analysis (Allison et al., 2006); in fact, bioinformaticians use clustering so often that researchers have compared clustering algorithms within the field (Yeung et al., 2001), including the problem of K-means initialisation. Those working within the field of computer vision also use K-means; an example of its use in this context would be to cluster entities in an image (Szeliski, 2010) using each pixel's features: normally their colour and position.

K-Means has also been applied to less scientific fields. Tsiptsis and Chorianopolous (2009:85) discuss its application in customer segmentation, providing a clear example of how it can be used in the industry, while Mucherino et al. (2009:47) present K-Means as a possible clustering solution for agricultural problems.

K-Means is so widely used that it is often pre-implemented in statistical or data analysis software, such as Matlab, SPSS, R, etc. Thanks to its ease of implementation it is often used in other commonly used software, such as Microsoft Excel.

2.4 Finding the initial centroids in K-Means

Automatic detection of the number of centroids is among the desirable properties of clustering algorithms (Callan, 2003: 313). Of course, in order to find the initial centroids, it is necessary to know also how many of them there are in a dataset. Finding the actual value of K and the initial centroids for K-Means is far from a trivial problem: the former part of it being, in fact, rather controversial. Some view the topic as baseless, because in many cases the value will depend on the viewing eye instead of the data and other factors, as discussed in Section 2.1 of this thesis. The problem of finding K and the centroids has been addressed in a number of publications, which generally speaking follow one of the following approaches:

(i) K-means is run a number of times with different values for K. Afterwards, a follow-up analysis determines the 'best' value for K and the specific centroids based on the optimisation of an index. Examples can be found in Hartigan (1975), Pelleg and Moore (2000), Tibshirani et al. (2001) and Monti et al. (2003).

(ii) The actual value of K and centroids is determined by evaluating the quality of clustering, using resampled versions of the data. Examples can be found in the work of McLachlan and Khan (2004), and references therein.

(iii) Use a hierarchical clustering procedure to determine the properties of

clusters and determine the values for K and centroids. Mojena (1977) and Pelleg and Moore (2000) provide examples of this.

Mirkin (2005:93) describes the 'intelligent' K-means (iK-Means) method, which belongs to the latter group and finds the number of clusters and their specific centroids using *anomalous* pattern (AP) clusters. In this method, the non-clustered entities located the farthest from the initial centre of gravity become, one at a time, tentative centroids. The cluster is then filled with all entities that are closer to this tentative centroid than to the centre of gravity itself. After all of the entities are clustered, the algorithm discards small clusters using a pre-specified threshold.

The method is intuitively appealing, as it is of easy interpretation even to those without a background in computer science or statistics. Computationally speaking, it adds little overhead especially if compared to the method in group (i) that needs to run K-means a number of times and group (ii) which needs to resample versions of data. Intelligent K-Means needs to be run only once, and it is a deterministic algorithm. Formally, the anomalous pattern algorithm can be defined in the following stages (Mirkin, 2005:91):

1. *Pre-processing*
   If the centre of gravity of the data is not defined, after standardisation set it to $a = (a_1, \ldots, a_M)$, where M is the number of dimensions. Assuming the Euclidian squared distance is used, the centre will be the grand mean of the dataset.
2. *Initial setting*
   Create a centroid, c, with the values of the entity that is the furthest from a.
3. *Cluster update*
   Create a cluster, S, of entities that are closer to c than to a. An entity y_i would be assigned to S iff $d(y_i, c) < d(y_i, a)$.
4. *Centroid update*
   If the centroid is not equal to the centre of gravity of cluster S, update it to the centre, and return to step 3.
5. *Output*
   Return the list S of entities and centroid c.

Algorithm 2.5: Anomalous pattern (AP) algorithm

In the AP algorithm, there is a version of K-means in which there are two clusters. The centroid of the first cluster is the centre of gravity a of the entire dataset itself, and the centroid of the second cluster has the values of the furthest entity from a. Anomaly in data can sometimes show as a cluster (Callan, 2003:314), and this idea is well aligned with the anomalous pattern approach of extracting clusters. Clearly, such clusters may demonstrate an ordinary pattern of the data or, possibly, a fault. In the latter case, the cluster is likely to become a singleton and be discarded.

Although AP clustering does not guarantee a global minimum for Equation (2.5), it works quickly and does provide a reproducible starting setting for K-Means (Stanforth et al., 2007). The intelligent K-means algorithm uses an anomalous pattern

iteratively to get one cluster at a time and can be formally defined in the following steps (Mirkin, 2005:93):

1. *Setting*
   Specify the *cluster discarding threshold* used to remove all anomalous clusters whose size is less than the threshold, and standardise the dataset.
2. *Anomalous pattern*
   Apply the anomalous pattern algorithm using the centre of gravity of the whole data $a = (a_1, \dots, a_M)$, where *M* is the number of dimensions. The position of this centre is not changed during any part of the algorithm.
3. *Control*
   Stop if all entities are clustered, otherwise remove the anomalous cluster found in the previous step from the dataset, and return to step 2.
   There are other possible stop conditions that may be useful, depending on the problem. These conditions are: reaching a pre-specified number of clusters (iK-Means being useful then to find the clusters and their centroids) if the first *t* clusters have a total contribution to the data scatter bigger than a pre-specified threshold, or if the contribution of a new cluster is too small.
4. *Removal of small clusters*
   Discard all clusters that are smaller than the pre-specified *cluster discarding threshold*. It is advisable to check the singletons, as there may be errors in the data, possibly data entry mistakes.
5. *K-Means*
   Run K-Means using the found centroids, and the full original dataset. Note this will effectively re-cluster the entities from the discarded clusters.

Algorithm 2.6: Intelligent K-Means

IK-Means is particularly efficacious when the clusters are well separated in the feature space (Mirkin, 2005:93), which is unsurprising, as in such a scenario it would be easier for the anomalous pattern part of the algorithm to find relevant clusters.

Mirkin (2005:110) also argues that iK-Means provides the user with a comfortable quantity of options for clustering, as it allows them to select options rather than having them imposed. These options are, for instance, to remove entities that are: (1) *deviant*, in that they belong to small, anomalous pattern clusters; (2) *intermediate* – entities that are far away from their centroids or have small attraction index values; (3) *trivial* – entities close to the centre of gravity of the dataset.

Intelligent K-Means has been tested in different publications. In a series of experiments, Chiang and Mirkin (2010) demonstrate that in most cases the original version of iK-Means outperforms many other K selection algorithms in terms of both cluster recovery and centroid recovery. These experiments use synthetic Gaussian clusters with between- and within-cluster spread to model cluster intermix.

Steinley and Brusco (2007) have further altered the intelligent K-Means algorithm to include a pre-specified K and no removal of singletons. Unsurprisingly, the results demonstrated in their experiments are below average. It seems sensible to discard such modifications when making further alterations to the algorithm to tailor it to specific problems.

A different, but also common, way of choosing a value for *K*, under the category (i) is discussed by Duda et al. (2001:557). Their suggestion is to run experiments with different numbers for *k*, where *k*=(1, ..., *K*), and analyse the results. The squared error criteria (Equation 2.5) will normally produce a smaller error for a larger *K*. If the data is intrinsically partitioned into $\hat{k}$ well-separated clusters, the error would decrease speedily until k=$\hat{k}$ and slowly afterwards. As one can imagine, the terms slowly and speedily are rather loosely defined, which may lead to difficulties; also, it may be necessary to perform too many experiments.

Metaheuristics are computational methods that may get deeper minima to the K-

Means criterion by iteratively improving a candidate solution, still without the guarantee of ever reaching an optimal solution. There are a number of such computational methods in clustering, including genetic algorithms (Goldberg, 1989; Maulik and Bandyopadhyay, 2000), particle swarm optimisation (Kennedy and Eberhart, 1995; Clerc and Kennefy, 2002) and variable neighbourhood search (Mladenovic and Hansen, 1997; Hansen and Mladenovic, 2001), but this subject is beyond the scope of this project.

2.5 Cluster validity

Cluster validity represents one of the main issues in clustering. The problem addressed here is: how can we know if a given cluster structure actually represents the data? This is a rather difficult question for a number of reasons:

(i) Clustering falls into unsupervised learning, which means that in most real-life scenarios the correct labels for the dataset will not be available. As a consequence, it is difficult to calculate an exact accuracy for the algorithm being used.

(ii) Different clustering algorithms may produce different cluster structures.

(iii) In most cases, the 'correct' number of clusters is unknown.

(iv) Even if there is no inner structure in the dataset, for instance if the dataset has a very high level of noise or is simply truly random, clustering algorithms will still find and output a structure.

For the latter problem, Smith and Jain (1984) suggest determining the c*lustering tendency* of the dataset. To avoid wasting time analysing clusters that do not represent the data structure, the *clustering tendency* should be found before any clustering algorithm is run on the dataset.

Cluster validity is also useful because if it is possible to quantify how close a clustering represents the data, then in principle it would be possible to automate the process of clustering data at different K and then analyse the structures. The area of cluster validity has received a considerable research effort and, in consequence, a number of possible indices can be used (Pedrycz, 2005:18; Dubes, 1987; Mirkin, 2005:232; Duda, 2001:557; Webb, 2002:396; Xu and Wunsch II, 2010:260).

There are three types of cluster validity indexes that may be used (Jain and Dubes, 1988:166): internal, external and relative.

2.5.1 Internal indices

An internal index attempts to quantify the relation between a cluster structure generated by any clustering algorithm and the data itself. The whole process of clustering and validation uses nothing but the data. Clearly, such indices can also help to determine the number of clusters (Pedrycz, 2005:18).

As pointed out by Mirkin (2005:233), silhouette width (Kaufman and Rousseeuw, 1990:96) is a popular measure that has shown good results (Pollard and Laan, 2002). It is defined for an entity $i \in I$ as:

$$sil(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))},$$

Equation 2.10: Silhouette width

where *a(i)* is the average dissimilarity of i with its cluster and *b(i)* is the smallest average dissimilarity of *i* from the other clusters: the larger the index, the more well clustered the entities.

2.5.2 Relative indices

A relative index compares different clustering structures generated by different clustering algorithms, or the same algorithm with different parameters, and helps to decide which of the structures is most appropriate.

A classical example is the Davies-Bouldin index (Davies and Bouldin, 1979). This index is well aligned with the K-Means criterion (Equation 2.8) as it attempts to maximise the distance between clusters and minimise the within-cluster distances to the centroids. As exemplified by (Xu and Wunsch II, 2010:269), it defines an index, R_k, which has the maximum comparison between cluster *k* and the other clusters.

$$R_k = \max_{j \neq k} \left(\frac{e_k + e_j}{D_{kj}} \right)$$

Equation 2.11: The maximum comparison between cluster *i* and the other clusters

in the above equation, D_{kj} represents the distance between the centroids c_k and c_j while e_k and e_j are their average errors. The Davies-Bouldin index can then be written as:

$$DB(K) = \frac{1}{K} \sum_{k=1}^{K} R_k$$

Equation 2.12: The Davies-Bouldin Index

Unsurprisingly, this type of index is data dependent, which means that different indices may provide very different outcomes. If choosing this type of index, it is advisable to run experiments with a set of them and then synthesize the results (Everitt et al., 2001; Xu and Wunsch II, 2010:271).

2.5.3 External indices

An external index measures the cluster structure against the correct labels for the dataset. Although it is unlikely that labels will be available, and if they are this would raise the question if clustering is the best approach to separate data, these indices tend to be very useful for algorithm evaluation. When designing a new algorithm it is not uncommon to test it with statistically generated data, such as Gaussian Mixtures Models, or labelled datasets. By doing so it is possible to have a better idea of what one can expect from them and maybe even the type of scenario in which the algorithm would excel. Any index under the other categories is not guaranteed, as the statistical problem of quantifying cluster validity is unfortunately unsolved (Duda, 2001:557). In this thesis we have chosen to use this type of index for the above reasons.

When using a clustering algorithm, the outputted labels will not necessarily match the known labels, even if they do correspond to each other. The issue is: the correct labels may 'name' a certain cluster S_k as *one* while the clustering algorithm may 'name' exactly the same cluster as *two*. That would not mean the clustering is incorrect, as *one* would correspond to *two*. Confusion matrices (also called matching matrices) are a common tool used to solve this issue. Such matrices normally have rows representing each cluster found with the clustering algorithm used; and columns representing the actual clusters; for example:

		Actual clusters		
		One	Two	Three
Found clusters	One	0	1	24
	Two	23	2	0
	Three	2	22	1

Figure 2.2 :Example of confusion matrix

In the above example, each cluster has 25 entities and it is possible to see a clear correspondence of cluster labels. For instance, the found cluster *one* corresponds to the actual cluster *three*, as there are 24 out of 25 entities clustered together. Confusion matrices are also useful for showing which clusters the algorithm is having difficulty separating. In order to quantify the evaluation of algorithms, we have used an intuitive measure called the Rand index (Rand, 1971):

$$R = \frac{a + b}{N},$$

Equation 2.13: Rand index

where *a* represents the number of entities that were clustered together and belong in fact to the same cluster, *b* the number of entities that were not clustered together and do not in fact belong to the same cluster, and *N* the total number of entities. In this scenario, the Rand index provides a rate representing the quantity of entities that were correctly clustered. In our results we multiplied the value by 100, for clarity.

2.6 Conclusion

In this chapter, we have demonstrated that there has been a considerable research effort applied over the years to clustering, and in particular to the K-Means algorithm. Such effort used clustering to solve the most diverse problems in both: academic and commercial environments. This has made K-means one of the most widely used partitional-clustering algorithms there is. Here we claim that although this is not the algorithm's only advantage, this fact alone means it is worthwhile evaluating it weaknesses and seeking its improvement.

We have discussed the advantages of K-Means that has led to its popularity, including its ease of implementation, speed, empirical success and even the fact that it is an intuitive approach. We have suggested that modifications to K-means can be easily adapted to other algorithms, such as PAM and K-median. In addition, we have stated that the lack of clustering re-visit in hierarchical clustering is a major disadvantage, as the iterative optimisation present in partitional clustering may gradually improve clusters.

In this chapter, we have acknowledged that one of K-Means' main weaknesses is that it does not clearly define an approach to identifying the number of clusters and centroids in a dataset. Mirkin (2005:93) has introduced an interesting algorithm called intelligent K-Means (iK-Means) to overcome this problem, as far as this is possible. Inevitably, the number of clusters will eventually depend on the resolution used to evaluate the number of clusters. The choice of iK-means out of the number of approaches discussed in the literature (Steinley and Brusco, 2007; Pelleg and More, 2000; Jain and Dubes, 1988) follows the successful experiments run by Chiang and Mirkin (2010) to demonstrate that, in most cases, iK-Means outperforms many other

K-selection algorithms in terms of cluster and centroid recovery.

Another major weakness of K-Means is the fact that it considers all features of a dataset to be of equal significance, despite the fact that in most real-world problems this may not be the case. We do not believe it is merely a matter of selecting the best features, as two different features, however significant, may be relevant for cluster separation at different degrees. We also understand that different features may have a varying degree of relevance at different clusters. We consider it then becomes vital to adjust the K-Means score function to such scenarios. This weighting should also take into consideration the classification bias generated by the Euclidian squared distance, as will be demonstrated in Chapter 3.

The next chapter of this thesis, 3, will suggest a compact solution to the above problems. In order to evaluate our algorithm, we have decided to use datasets whose labels are known; this has allowed for the preparation of a confusion matrix for each clustering and, therefore, a Rand index.

Chapter 3: Feature weighting at K-Means clustering

3.1 Feature selection and weighting

Classification, via clustering or otherwise, requires the formulation of a hypothesis. The latter is a pattern or function used to predict classes based on given data (Liu and Modota, 2008). The size of the hypothesis space is directly proportional to the number of features in the data; more specifically, as argued by Liu and Modota, a linear increase in the number of features leads to the exponential increase in the hypothesis space, otherwise known as the *curse of dimensionality* (Bellman, 1957). The issue we face here is that a dataset with a large number of features may become too difficult for a given algorithm to work on. The smaller the number of features, the smaller the hypothesis space and by consequence the easier it is for a given algorithm to find the best hypothesis.

In the case of K-Means based clustering, we face a further issue. The K-Means criterion (Equation 2.8) uses a distance function to determine how similar entities are, and decide if they should be put into the same cluster. The precision of a distance function is then crucial to the criterion but it may be lost with an increase in the number of dimensions, in which case the difference between the nearest and the farthest entity may become meaningless (Beyer et al.,1999).

$$\lim_{d\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0$$

Equation 3.1: Loss of meaning between the farthest and nearest entities with the increase of dimensions

Feature selection may help to track the problems cited above, by reducing the number of features in the dataset (Dash et al., 2002, Friedman and Meulman, 2004; Raftery and Dean, 2006). It aims to discard those that are redundant or least information carrying (Pal and Mitra, 2004:59; Liu and Modota, 2008). Features carrying the least information may mislead an algorithm as they may be given the same degree of relevance as information-carrying features. Redundant features, which are a set of two or more features, each of them being relevant but carrying the same information, are unlikely to mislead a classifier. Of course, it is possible to face a number of such sets of redundant features in the data. In this context, we would like to keep only one of the features per set, so the actual information is not lost. Our aim regarding the latter is to reduce computational effort as a given algorithm would deal then with a smaller feature space and avoid a loss of meaning in the distance function.

There is a further solution to this problem, known as feature extraction. This type of algorithm follows a different approach for dimensionality reduction. Instead of simply selecting which features should be used it incorporates them all to create a new and reduced set of them. Although these new features have a relationship to the original ones, they are not the same but a transformed version, as achieved for instance by principal component analysis (PCA), a well-known feature extraction technique. However, this type of algorithm is out of the scope of this work.

Dy (2008) finds that feature selection algorithms normally have two main components:

(i) *Feature search*

The search strategies attempt to find an optimal subset of features that will provide greater class separability. There are a number of examples of these strategies. Whitney (1971) introduces a common example, *sequential forward selection*, in which the

subset of the selected features is empty at the start, with new features selected one at a time until one is added that does not improve the subset quality. Whitney was likely to have been inspired by an earlier algorithm introduced by Marill and Green (1963), known as *sequential backward selection*, which intuitively works in the opposite fashion. In this, the subset of selected features is initialised with the original set's features; the algorithm then removes one feature at a time until removing another feature does not improve the subset quality.

However, these classical strategies may become too computationally intensive in some situations and may not be the best approach when there are *interacting features* in the dataset. These type of features usually defy the above heuristic solutions because the information they carry cannot be detected on an individual basis (Liu and Motoda, 2008). The logical XOR (Guyon and Elisseeff, 2003), is a common example of this, as both features are necessary to define the class and only one would be useless.

(ii) *Feature evaluation*

Each selected feature or subset of selected features needs to be evaluated with a relevance criterion. In the *sequential forward selection* and *sequential backwards selection* algorithms, it is necessary to evaluate a subset of features in order to decide if more features should be added or removed. In these algorithms, this involves taking into account which features contribute the most and the least to the relevance criterion function.

Feature evaluation is especially difficult in clustering, as there are no labelled entities available to be used to find a relation between feature(s) and class(es). It is in fact the decision of choosing what criterion to use that makes feature selection in clustering difficult (Dy, 2008).

Feature selection algorithms, as a whole – feature search and feature evaluation, being these supervised or not, can be categorised in different ways. Arguably, the most common way of categorising them is based on their dependency of the feature evaluation method on the learning algorithm. There are then, two primary categories:

(i) *Filter methods*

Feature selection algorithms of this type do not depend on the learning algorithm in use. They use the properties of the data itself to select which features should be kept without taking into account the classification or clustering algorithm(s) that will be subsequently applied to the dataset. The method name is intuitive as the idea is to *filter* the data before classification or clustering is applied.

(ii) *Wrapper* methods

In this method, the feature selection algorithms use the learning algorithm to determine the quality of the subset of the selected features. If clustering is used, a wrapper method would incorporate the feature selection algorithm inside the chosen clustering algorithm.

In this work, we have chosen K-Means as the initial clustering algorithm to be enhanced. As Dy (2008) argues that for a particular algorithm a wrapper approach may produce better results than the filter approach, it seems sensible for us to follow this approach to feature selection. However, Dy also acknowledges that such an approach may be more computationally expensive, as there are multiple runs for different feature subsets. It is worth noting that the number of clusters, and by consequence the quantity and value of centroids, will depend on the feature subset.

Another point of interest is that ordinary feature selection follows a binary approach, in which features are either selected or not selected. Normally, all selected

features are then considered of equal relevance in clustering. We acknowledge here that this may not always be the best approach and that selected features may have a different degree of relevance when clustering, or in fact, when using any algorithm for pattern recognition. We follow then a generalisation of feature selection, in which the features' participation is not decided in a binary fashion but in a range [0, 1]. Features are no longer 'selected' but 'weighted'. Following this approach does not necessarily cancel the feature selection mode, as if two features do have the same degree of importance, this can be achieved by them having the same weight.

Taking into account the fact that K-Means is an unsupervised algorithm, feature weighting should follow such a learning method or require a very limited amount of labelled data: another reason for this is that such data can be expensive to acquire or subjective to the labeller.

The goal of feature selection for unsupervised learning is to find the smallest feature subset that best separates 'interesting natural' clusters from data (Dy, 2008; Dy and Brodley, 2004). The definition in no way limits the subset of features to be the same to all clusters, which leads us to a further way of categorising feature selection algorithms. These are *global* in the sense that the same subset of selected features is valid for all clusters, or *local* in which the subset of selected features may be different at different clusters. Local methods in clustering, also called *subspace clustering*, were introduced by Agrawal et al. (1998). These are intuitively appealing, as in certain scenarios different clusters may indeed be easily separated with a different subset of features; also, it does not cancel a global approach, as a local method may find that the same subset of features works fine with all clusters. Furthermore, the general idea of such methods can be easily generalised to feature weighting, in which features' weights may vary at different clusters.

Feature selection or weighting algorithms may, at times, just like K-Means itself, get trapped in local optima or be too complex or computationally demanding. A possible solution for these scenarios is to use a randomized algorithm, of which there are two basic types (Stracuzzi, 2008): *Las Vegas* and *Monte Carlo*. The former always outputs the correct result but may take a long time to do so, while the latter is always fast but may output an incorrect answer. Randomized algorithms commonly provide initial solutions fast and improve them overtime, in a similar fashion to *anytime* algorithms (Zilberstein, 1996). Genetic algorithms are a general-purpose method for randomized searches. As our base algorithm, K-Means, suffers from this problem itself, it becomes rather difficult to track the issue in a wrapper-based feature weighting algorithm, and we will leave this for further research.

3.2 Feature weighting in K-Means

The primary concern of this chapter is that despite its popularity, K-Means may have difficulties in clustering data with irrelevant, or noise features. The general idea would then be to include feature weighting in the K-Means clustering criterion (Equation 2.8), which revert us to the issue of distance definition. Feature weighting is in fact part of the definition of the distance measure one utilizes in K-Means or other distance-based clustering algorithms. The majority of work in clustering uses Euclidean squared distance (Equation 2.1) to find the dissimilarity between multidimensional entities. We can see three main approaches for extending this distance measure: (1) extension of the inner product in the distance definition; (2) nonlinear weights, and (3) the use of other distance measures.

(1) Extending the inner product in the distance definition

This is equivalent to using linear combinations of the original features. It involves a semi-positive definite weight matrix *W*, so that $d(x,y) = (x-y)^T W(x-y) = \sum_{ik} w_{ij}(x_i - y_i)(x_j - y_j)$, where the superscript *T* denotes transpose; see, for example, Xing et al. (2003) and Bilenko et al. (2004). In this framework, there is a special case in which the weight matrix W is diagonal; in such a case, $d(x,y) = (x-y)^T W(x-y) = \sum_i (x_i - y_i)^2$ and the feature weights are also feature-rescaling factors of $\sqrt{w_i}$: in this last scenario, the distance measure is defined and performs feature selection through feature weighting, and the issue of data normalisation is also addressed.

In this linear approach, the weights are yet data-driven but also based on a different criterion for clustering (see, for example Modha and Spangler, 2003; and Tsai and Chiu, 2008) or additional information of the cluster structure to be found (see Strehl et al., 2000; Xing et al., 2003; and Huang et al., 2008). Strehl et al. and Xing et al. take a different approach regarding the measurement of the algorithms' accuracy, as they both use a pairwise approach in which paired entities are either in the same cluster or not, while Huang et al. use cluster labels on all entities. In Huang et al.'s case, using the ordinary K-Means criterion would select a single feature, the one with the smallest within-cluster variance, yet this is unlikely to be an optimal solution in most real-world scenarios. Taking this issue into account, Modha and Spangler develop an interesting method that makes major changes to the K-Means criterion. Their method presents a division of the 'within-cluster dispersion' over an index representing the 'between-cluster dispersion'. They evaluate their method in measures of cluster precision and recall, with 83% accuracy in the Australian credit-card dataset (to be introduced in Section 3.4). This can be considered as suboptimal if compared with Weighted K-Means and Intelligent Minkowski Weighed K-Means, which have a maximum

accuracy of 85% (Huang et al., 2008) and 86.09%, respectively, for the same dataset, as we show in Table 3.10; these algorithms are discussed in Section 3.2.1 and Section 3.3.3, respectively. In fact, their lower accuracy does not come as a surprise. While the two latter algorithms assign one weight per feature in each cluster subspace, Modha and Spangler assign only two weights pertaining to the subspaces of quantitative and categorical features.

Strehl et al. (2000) and Xing et al. (2003) also modify the K-Means criterion. They use their pairwise lists to impose constraints in clustering entities together or separately and add a penalising procedure if these constraints are broken.

There have been other researchers who have sought not to change the K-Means criterion. Tsai and Chiu (2008) introduce a method which also iteratively adjusts the weights, and experiment with the iris and wine datasets. It is worth commenting that in their experiments they use the adjusted Rand index which is not the exact same index we use here (Rand Index), so one should expect lower values to occur in their experiments. Even so, the *correspondent* values of accuracies of 0.77 and 0.40 by Tsai and Chiu (2008:4667), using the adjusted Rand index with the two datasets, are considerably below those that we show in Table 3.1 and Table 3.4.

(2) Non-linear weights

The non-linear weights considered so far, are weight powers which express the distance as $d(x,y) = (x-y)^T W^\beta (x-y) = \sum_i w_i^\beta (x_i - y_i)^2$. In this, W is a diagonal feature weight matrix and T the transpose. To our knowledge, non-linear weights were first used by Makarenkov and Legendre (2001), at β=2 in their three-stage extension of K-Means, which minimises the K-Means summary distance criterion as a function of

three groups of variables: entity memberships, cluster centroids, and feature weights. It is worth mentioning that at β=2, the weights become feature-rescaling factors.

At first, the above method may seem flawed, because as shown by Huang et al. (2008) it would result in selecting a single feature by setting its weight to one and disregarding all others; however, this is not the case. The solution comes from the squares at the weights. A simplified illustrative example is: a combination $w_1n_1+w_2n_2$ of numbers n_1=1 and n_2=3 would produce a minimum of 1, over the constraints $w_1 \geq 0$, $w_2 \geq 0$, and w_1+w_2=1, obviously at w_1=1. However, the combination $w_1^2n_1+w_2^2n_2$ involving quadratic weights under the same constraints, reaches its minimum of ¾, well inside the boundaries, at $w_1=3/4$ and $w_2=1/4$.

Cluster specific feature weights were introduced, to our knowledge, by Frigui and Nasraoui (2004). They also do not go further than β=2 and have changed the K-Means criterion as well by introducing 'regularisation' terms.

Chan et al. (2004) and the extension of their method to address cluster specific weights shown in Huang et al. (2005, 2008), further extend the K-Means criterion. They introduce an arbitrary exponent β but utilise yet the squared Euclidean distance measure (Equation 2.1). They also measure the success of their algorithm in a different way; they do not use the summary distance to the centroid but rather intuitively the recovery of clusters against their labels. Also, it is worth noting that it would not be possible to compare the values of the summary distance criterion at different values for β. They also show that most β values are data specific, but unfortunately do not propose a method to find such values when labels are unknown beforehand. Another issue that we intend to address in this chapter is that when $\beta \neq 2$, the weights cease to be feature-rescaling factors; to address this, we generalise the Euclidean squared distance they use to the Minkowski metric of the same power.

Green et al. (1990) summarise much of the effort produced to their publication date in non-linear feature weighting, including the early work of DeSarbo et al. (1984). A continuation of the formers' work was undertaken more recently by Makarenkov and Legendre (2001), who presents an optimal variable weighting method (OVW) also based on the K-Means criterion using squared Euclidean distance. Their method has feature weights with rescaling coefficients which makes it equivalent to Minkowski Weighted K-Means, which we present in Section 3.3.2, at β=2. Makarenkov and Legendre also anticipate a Minkowski Weighted K-Means method by stating: "The development of an optimal weighting algorithm for Minkowski metric is an interesting topic for further investigation".

(3) Minkowski metric and other distances

Equation 2.3 shows the Minkowski p-metric definition. This metric has already found its use in location analysis (Chakerian and Ghandehari, 1985; Revelle and Eiselt, 2005; Lawrence, 2006), but its use in clustering is still limited.

Traditionally, most research effort in clustering uses the Euclidean metric, the Hamming (also known as the city-block and the Manhattan) and the Chebyshev (or the maximum metric), the equivalents of the Minkowski metric at β=2, β=1 and $\beta \rightarrow \infty$, respectively; however, recently a few others have considered the use of other values of p in clustering. In terms of normalisation, Doherty et al. (2004) note that cluster recovery improves differently with K-Means at different p values. The effects of data concentration at different p values have been discussed by Rudin (2009) and Francois et al. (2007), who also cite the possibilities of utilising them for tackling data analysis problems; the latter paper mentions a number of useful mathematical properties of the

Minkowski metric. The Minkowski *p*-metric has been already incorporated to the least-squares approach, in prediction by Kivinen et al. (2006),who point to the fact that, generally speaking, the metric allows us to identify and, in fact, ignore irrelevant features, in line with the main aim of this chapter.

Although Banerjee et al. (2005) introduce a number of different metrics; it seems to us that a comparative analysis of different metrics do not attract much attention. Of course, there are exceptions to this rule, such as web-page clustering (see, for example, Strehl et al., 2000).

3.2.1 Weighted K-Means

The Weighted K-Means algorithm (abbreviated as WK-Means) described by Chan, Huang and their collaborators (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008) is a modification of the K-Means criterion (equation 2.8) to include unknown weights, w_v to the features v, v=1,..., M in the dataset. Their approach to the problem relates feature weights to a set of patterns during the process of clustering, aligned to the wrapper approach idea for feature selection.

$$W(S, C, w) = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} s_{ik} w_v^{\beta} (y_{iv} - c_{kv})^2,$$

Equation 3.2: Weighted K-Means criterion

subject to:

$$\begin{cases} s_{ik} \in \{0,1\} \\ \sum_{k=1}^{K} s_{ik} = 1 \\ \sum_{v=1}^{M} w_v = 1 \end{cases}$$

According to the criterion, each feature weight w_v should be non-negative and add to the unity. The criterion has a user-defined parameter β, which expresses the rate of impact weights will have on their contribution to the distance. The WK-Means algorithm follows an iterative optimisation similar to K-Means, and by consequence it is affected by some of its strengths, such as its convergence in a finite number of iterations, and its weaknesses, which we shall discuss in further detail shortly in this section.

WK-Means presents an extra step in relation to K-Means, as it updates the feature weights according to Equation 3.3:

$$w_v = \frac{1}{\sum_{u \in V} [\frac{D_v}{D_u}]^{\frac{1}{\beta-1}}},$$

Equation 3.3: WK-Means weight update

where D_v is the sum of within-cluster variances of feature *v* weighted by clusters' cardinalities:

$$D_v = \sum_{k=1}^{K} \sum_{i \in I} (y_{iv} - c_{kv})^2 s_{ik}$$

Equation 3.4: Within-cluster variances of feature *v*

As stated in the literature (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008), there are two scenarios in which Equation 3.3 is not applicable. These are:

(i) when $D_u = 0$ for a $u \in$ V

Note that in such an instance, there would be an undesirable division by zero. To solve the issue they suggest adding a non-negative constant to each item in the definition of D_u, such as the average dispersion of all features in the dataset.

(ii) when β=1

Note that this would also generate a division by zero. According to the literature on WK-Means (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008), the minimum of its criterion (Equation 3.2) is then reached by setting $w_{v*} = 1$ where *v** represents the feature with the smallest sum of within-cluster variance; all other feature weights are set to 0.

The distance measure needs to be adjusted to take into account the weights found in this algorithm. The Equation 3.5 below shows the distance between the feature *v* of an entity y_i and the centroidc_k.

$$d(y_i, c_k) = \sum_{v \in V} w_v^{\beta} (y_{iv} - c_{kv})^2$$

Equation 3.5: The adjusted distance measure to take the weights into account

The main steps of WK-Means are shown in algorithm 3.1, below.

1. *Initial setting*
   Similar to K-Means, this method uses an external parameter to define the number *K* of clusters. WK-Means then randomly initialises the centroids and the feature weights, ensuring that they add up to unity.
2. *Cluster update*
   The cluster update assigns each entity to their closest centroid, using the adjusted distance measure that takes into account the feature weights; this is shown in Equation 3.5.
3. *Stop condition*
   This is as per the K-Means algorithm.
4. *Centroids update*
   Updates centroids to the centre of gravity of its cluster. Again, as the method is based on Euclidean distance, this is achieved by moving the centroids to the mean of their clusters.
5. *Weights update*
   Updates the weights according to Equation 3.3, constrained by the condition that the sum of weights should be one.

Algorithm 3.1: Weighted K-Means

As intuition indicates, the optimal weight of a feature may be cluster-dependent. A feature *v* may have a high degree of relevance within one cluster *k*, but a smaller degree in a different cluster *k**. Huang (2005) introduces this characteristic to WK-Means under the name *subspace clustering* (Agrawal et al., 1998). This is a rather easy extension to the above, as all of the equations have only minor adjustments: w_v and D_v become w_{vk} and D_{vk} respectively, where k=1, ..., K.

Because of its similarity to K-Means, WK-Means inherits some of the former's weaknesses: for instance, the algorithm yet initialises the centroids randomly, not guaranteeing an optimal solution. Also the same applies to the weights, creating the possibility that these could be far from representing the relevance of the features.

3.3 Extensions of Weighted K-Means

In this section, we present extensions to Weighted K-Means. Apart from the obvious aim of increasing its cluster recovery and resistance to noise features, we also hope to generalise its distance measure by using the Minkowski metric and to transform the feature weights in feature rescaling factors for $p \geq 1$.

3.3.1 Intelligent Weighted K-Means

Our first step was to extend the Weighted K-Means algorithm, hoping to supply it with good initial centroids, instead of random ones. The anomalous cluster initialisation provided by intelligent K-Means (Mirkin, 2005:93) proved easy to integrate into the WK-Means algorithm (Algorithm 3.1). An interesting point to note is that the fundamentals of this pre-initialisation can be expanded to find initial weights for features, so the weights also cease to be initialised randomly. We have called this algorithm intelligent Weighted K-Means (abbreviated as iWK-Means); it can be described with Algorithm 3.2:

1. *Initial setting*

   Each feature's weights are initialised with equal values and add up to unity. This algorithm uses an external parameter to define the minimum cluster size; the value for this is, typically, two.

2. *Cluster update*

   The non-clustered entity farthest away from the centre of gravity of the whole dataset, the origin 0, is taken as the tentative anomalous cluster's centroid.

   a. Define a cluster *S* to consist of the entities that are closer to the tentative centroid than to the origin 0. The distance measurement takes the weights into account, using Equation 3.5;

   b. Update the centroid to the centre of gravity of *S*, its mean*;*

   c. Update the weights for this centroid, using Equation 3.3;

   d. If the new centroid differs from the previous one or is the first centroid for *S*, return to step a; otherwise, stop and remove *S* from the dataset.

3. *Cluster all entities*

   Repeat step 2 until all entities are clustered.

4. *Discard less significant clusters*

   Remove all centroids whose cluster size is smaller than the initially specified threshold.

5. *Run WK-Means*

   Run WK-Means starting with the found centroids and weights.

Algorithm 3.2: Intelligent Weighted K-Means (iWK-Means)

3.3.2 Minkowski metric Weighted K-Means

If we generalise the distance from the WK-Means criterion (Equation 3.2) from squared Euclidean to Minkowski distance, we will be able to associate the feature weights

found by WK-Means with feature-scaling factors for any $\beta \geq 1$.

$$W_\beta(S, C, w) = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} S_{ik} w_v^\beta |y_{iv} - c_{kv}|^\beta = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} S_{ik} |w_v y_{iv} - w_v c_{kv}|^\beta$$

$$= \sum_{k=1}^{K} \sum_{i \in S_k} d^\beta(y'_i, c'_k),$$

Equation 3.6: Minkowski metric Weighted K-Means criterion

subject to:

$$\begin{cases} s_{ik} \in \{0,1\} \\ \sum_{k=1}^{K} s_{ik} = 1 \\ \sum_{v=1}^{M} w_v = 1 \end{cases}$$

The difference between the WK-criterion (Equation 3.2) and the above Minkowski Weighted K-Means (abbreviated as MWK-Means) is the distance exponent which has been generalised from 2 into β: this expression then refers to the β power of the Minkowski distance between the rescaled points $y_i' = (w_v y_{iv})$ and centroids $c_k' = (w_v c_{kv})$ in the feature space. For reference, the Minkowski distance is defined in Equation 2.3.

The MWK-Means criterion above is a weighted version of K-Means that identifies weights with feature rescaling coefficients, this is possible because the two βs in the criterion are the same. We avoid problems with degeneration by setting entities as initial centroids, and by looking for relatively small number of clusters.

The Minimization of the MWK-Means criterion is rather complex and cannot be easily solved. We take then an alternating minimization approach over the three groups of variables: clusters, centroids and weights, similar to the one utilized to minimize WK-Means (algorithm 3.1). Any necessary change happens solely because of the exponent β.

When using Euclidean distance, the centre of gravity of data is defined as its mean; however, here it should be the Minkowski centre. When using Euclidean distance, the centre of gravity is the point with the smallest average distance to all entities in a cluster; when using the Minkowski metric, this scheme is not different but ceases to be the average of the cluster. The point is to minimise the Minkowski distance, or equivalently its β power, to the cluster's entities. In MWK-Means this is given by:

$$d(c) = \sum_{i \in S_k} w_v^{\beta} |y_{iv} - c|^{\beta}$$

Equation 3.7: Minkowski centre in MWK-Means

In Algorithm 3.4, we show an exact procedure to find the centre as per the above equation. In terms of weight update, this now has to take into account the β, and is defined then as:

$$w_v = \frac{1}{\sum_{u \in V} \left[D_{v\beta} / D_{u\beta} \right]^{\frac{1}{\beta - 1}}},$$

Equation 3.8: Weight update in MWK-Means

where the dispersion per feature is:

$$D_{v\beta} = \sum_{k=1}^{K} \sum_{i \in I} |y_{iv} - c_{kv}|^{\beta} s_{ik}$$

Equation 3.9: Dispersion in MWK-Means

To prove that Equation 3.8: *Weight Update in MWK-Means* gives the optimal weights, given the centroids and clusters, one should follow analogous proofs in Chan et al. (2004) and Huang et al. (2005). Specifically, by using $D_{v\beta}$ defined in Equation 3.9: *Dispersion in MWK-Means*, the Minkowski criterion can be rewritten as $W_{\beta}(S,C,w)=\sum_{v=1}^{M} w_v^{\beta} D_{v\beta}$. To minimize this with regard to the constraint $\sum_{v=1}^{M} w_v = 1$, one should use the first-order optimality condition for the Lagrange function $L = \sum_{v=1}^{M} w_v^{\beta} D_{v\beta} + \lambda(1 - \sum_{v=1}^{M} w_v)$. The derivate of L with respect to w_v is equal to $\partial L/\partial w_v = \beta w_v^{\beta-1} D_{v\beta} - \lambda$. By equating this to zero, one can easily derive that $(\lambda/\beta)^{\frac{1}{\beta-1}} = w_v D_{v\beta}^{\frac{1}{\beta-1}}$, that is, $w_v = (\lambda/\beta D_{v\beta})^{\frac{1}{\beta-1}}$. By summing these expressions over all v, one arrives at equation $L = \sum_v (\lambda/\beta D_{v\beta})^{\frac{1}{\beta-1}}$, so that $(\lambda/\beta)^{\frac{1}{\beta-1}} = 1/\sum_v (1/D_{v\beta})^{\frac{1}{\beta-1}}$. This leads to Equation 3.8 indeed.

As with the former equation, this does not apply when a feature in the dataset has a dispersion of zero, and the issue is solved in the same manner by adding a small constant. We also follow Huang et al. (2005, 2008) solution for β = 1, selecting a single feature. A formalisation of the differences between MWK-Means and K-Means in steps 2, 4 and 5, at a given β, follows:

2 *Cluster update*

Given a set of centroids and a set of weights per feature and cluster, update the clusters by using the minimum distance rule with the distance defined as β power of Minkowski distance in 2.3.

4 *Centroid update*

Given a set of clusters and a set of weights per feature and cluster, update the centroid of each cluster as its Minkowski centre, as per Equation 3.7: Minkowski centre in MWK-Means.

5 *Weights update*

Given a set of clusters with respective centroids, update the weights according to Equation 3.8, constrained by the condition that the sum of the weights is one.

Algorithm 3.3: Minkowski Weighted K-Means (MWK-Means)

For the above algorithm to be practical we need a computationally feasible procedure to obtain the Minkowski centre of the features at any $\beta \geq 1$. Note that each feature is in fact a set of reals y_i, $i=1,2,\ldots, N_k$, where N_k is the number of entities in the cluster k. In the space of $\beta \geq 1$, as per Equation 3.7, the Minkowski centre in MWK-Means is a convex function of c, allowing us to propose minimising it using a steepest-descent procedure. Another point to consider is that the minimum for β=1, which is equivalent to the Hamming distance (Equation 2.2), is known to be the median of y_i, so our minimising method only needs to consider the cases in which β>1.

We assume that the values in y_i are sorted so that $y_1 \leq y_2 \leq \ldots \leq y_{Nk}$, the ascending order. Firstly, we prove that the minimum of Equation 3.7 c is in the interval of $[y_1, y_{Nk}]$. If the minimum were to be reached outside the interval, for instance at $c > y_{Nk}$, then $d(y_{Nk}) < d(c)$ because $|y_i - y_{Nk}| < |y_i - c|$ for all $i=1,2, \ldots, N_k$. The same would hold for the β-th powers of these. This is a contradiction, and by consequence proves

our initial assumption of c being in the interval f [y_1, y_{Nk}].

In order to show the convexity, we consider any c in the interval [y_1, y_{Nk}]. Equation 3.7 can then be re-written as:

$$d(c) = \sum_{i \in I^+} (c - y_i)^\beta + \sum_{i \in I^-} (y_i - c)^\beta,$$

Equation 3.10: Minkowski centre in MWK-Means, re-written

where I^+ represents those indices i=1, 2, ..., N_k in which $c > y_i$ and I^- those in which $c \leq y_i$. Then, the first derivative of $d(c)$ can be expressed as:

$$d'(c) = \beta(\sum_{i \in I^+} (c - y_{i)}^{\beta-1} - \sum_{i \in I^-} (y_i - c)^{\beta-1})$$

Equation 3.11: First derivative of *d(c)* equation 3.10

and the second derivative as:

$$d''(c) = \beta(\beta - 1)(\sum_{i \in I^+} (c - y_i)^{\beta-2} + \sum_{i \in I^-} (y_i - c)^{\beta-2})$$

Equation 3.12: Second derivative of *d(c)* equation 3.10

for $\beta > 1$, the above equation will always have a positive value: this proves it is convex.

The convexity of the equation for $\beta > 1$, the only interval we will use, leads to another useful property: assume $d(y_{i*})$ is the minimum for every $d(y_i)$ value (i=1, 2, ...,

N_k) and denote by $y_{i'}$ that the y_i-value, which is the nearest to y_{i*} among those smaller than y_{i*}, at which $d(y_i)<d(y_{i*})$. Similarly, denote by $y_{i''}$ that y_i-value, which is the nearest to y_{i*} among those that are greater than y_{i*}, at which $d(y_i)>d(y_{i*})$. Then, the minimum of $d(c)$ lies within the interval $[y_{i'}, y_{i''}]$.

The above properties validate the following steepest-descent algorithm, which we use to find the Minkowski's centre of a set $\{y_i\}$ of reals $y_1 \leq y_2 \leq \ldots \leq y_{Nk}$ at $\beta>1$:

1. *Initialisation*

   Initialise with $c_0=y_{i*}$, the minimiser of $d(c)$ on the set $\{y_i\}$and a positive learning rate λ that can be taken, as approximately 10% of the difference $y_{Nk}-y_1$.

2. *Update c1*

   Compute $c_0 - \lambda d'(c_0)$ and take it as c_1 if it falls within the interval $[y_{i'}, y_{i''}]$. Otherwise, decrease λ by approximately 10%, and repeat this step.

3. *Stop condition*

   Test whether c_1 and c_0 coincide with a pre-specified threshold. If they do, halt the process and output c_1 as the optimal value of c. If not, move on to the next step.

4. *Update c_0*

   Test whether $d(c_1) \leq d(c_0)$. If it is, set $c_0=c_1$ and $d(c_0)=d(c_1)$, and go to (2). If not, decrease λ a by approximately 10%, and go to (2) without

Algorithm 3.4: Finding the Minkowski centre with the steepest-descent procedure

In this algorithm we find one coordinate centre at a time, reducing the problem to one dimension only. We do this because to minimize W_β over $c_{kv} = \sum_{k\,v} W_\beta(k,v)$ one minimizes each $W_\beta(k,v) = \sum_{i\epsilon\, I} s_{ik} w_v^\beta |y_{iv} - c_{kv}|^\beta$ separately.

3.3.3 Intelligent Minkowski metric Weighted K-Means

In this section, we introduce a final algorithm. It uses anomalous clusters found by using Intelligent K-Means and the Minkowski distance to find the initial centroids and weights. We call this algorithm intelligent Minkowski Weighted K-Means (iMWK-Means).

Through similar experimentation, we found that adding a very small constant to the Equation 3.4: *Within-cluster variances of feature v*, such as 0.01, instead of the average feature variance, as per Huang et al. (2005, 2008) and Chan et al. (2004) tends to increase accuracy. Note that we only do this in the first, 'intelligent', step of the algorithm: MWK-Means remains unchanged.

3.4 Setting of the experiment

We have introduced three K-Means based algorithms and described further three, including K-Means itself. It seems reasonable to conduct experiments with all of these in order to get a formal comparison. Our experiments follow two directions:

(1) Test our proposed Minkowski metric based algorithms for cluster recovery, especially in datasets with a large amount of noise features, and compare them with the original WK-Means described by Huang et al. (2005, 2008).

(2) Explore the behaviour of our proposed methods, and indeed the original WK-Means with respect to increasing the number of features.

The algorithms used for the experiments are each run with a pre-specified number, *K*, of clusters; these are:

1. K-Means (K-M): we show the result of a hundred runs of generic K-Means at random initialisations of centroids;

2. WK-Means (WK-M): we show the results of a hundred runs of the Weighted K-Means algorithm by Huang et al. (2005, 2008);

3. MWK-Means (MWK-M): we show the results of a hundred runs of the Minkowski metric Weighted K-Means;

4. iK-Means (iK-M): this is the generic K-Means initialised at the centroids of *K* largest anomalous clusters, as described by Mirkin (2005:93);

5. iWK-Means (iWK-M): we show the result of the Weighted K-Means algorithm initialised at the centroids and weights of the *K* largest anomalous clusters;

6. iMWK-Means (iMWK-M): we show the result of the Minkowski metric Weighted K-Means algorithm initialised at the centroids and weights of the *K* largest anomalous clusters.

In the weighted algorithms among the above (Algorithms 2, 3, 5 and 6), a given weight depends on a feature and cluster, so one feature may have different weights at different clusters.

To facilitate comparison, all datasets used in our experiments are cluster labelled. This allows us to make a clear comparison of the algorithms' accuracy based on the Rand Index. In order to do so, we prepare a confusion matrix in a very similar way to Huang et al. (2008); we map each *k* cluster *k=1, ..., K*, produced by the algorithms to a pre-labelled *K* cluster, one at a time from the largest overlap to the

smallest. All of the overlaps are then merged to calculate the proportion of correctly labelled clusters. As they are not comparable at different β's, we did not use the values of the K-Means criterion itself.

It should be noted that the above algorithms were designed for clustering rather than classification. Following the work of Chan et al. (2004) and Huang et al. (2005, 2008), in our setting, only β, a single parameter which is the feature-weight exponent, and in the case of the Minkowski versions a feature-rescaling factor, is subject to adjustment to the pre-specified class labels. This is rather different from the adjustments of classifications algorithms in which the number of parameters to be adjusted always exceed the number of features. An interesting property of this is that there will likely not be much overfitting, which can happen when a classification algorithm is run. In this chapter, we find those values of β at which each algorithm performs best, following the work of Chan et al. (2004) and Huang et al. (2005, 2008).

Next, we describe the synthetic and real datasets used in our experiments. All of the datasets are labelled, as this is a requirement of the Rand Index, as justified in Chapter 2, Section 2.5.3.

(1) Real datasets

The real datasets were all acquired in the UCI Machine Learning Repository (Asuncion and Newman, 2007). These are:

(i) Iris dataset, comprised of 150 flower specimens, each with features: sepal length, sepal width, petal length and petal width; and partitioned into three clusters regarding their species: Setosa, Virginica and Versicolor.

(ii) Wine dataset, containing the results of chemical analyses of 178 specimens

of wine with 13 features, partitioned into three clusters as per their region of production.

(iii) Pima Indians diabetes dataset, containing the data of 768 females aged 21 years and over of Pima Indian heritage. The dataset has eight numerical features and is partitioned into two clusters representing the results of these women's positive and negative tests for diabetes.

(iv) Hepatitis dataset, containing 155 specimens with 19 features of numerical and categorical data, partitioned into two clusters representing the status of the subject as alive or deceased.

(v) Australian credit-approval dataset, containing 690 specimens of credit applications with 14 categorical, integer and real features. The dataset is partitioned into two clusters representing the approval or rejection of a credit application.

(vi) Heart disease dataset, containing 690 specimens divided into 13 categorical and real features. The dataset is partitioned into two clusters representing the presence or absence of a heart condition.

Datasets (i), (ii), (iii) and (iv) were chosen because of their wide use in the literature, which makes it easier to compare our method with others'. The latter two, (v) and (vi), were chosen because experiments with these datasets were present in the latest publication regarding WK-Means (Huang et al., 2008) facilitating the comparison of our method with theirs. Also, some of the datasets have either a rather complex class structures, or clusters are not well separated. For instance none of the existing classification algorithms have achieved greater than 70-80% accuracy on the Pima Indians diabetes dataset.

(2) Synthetic datasets

We have also experimented with synthetic datasets and generated the following:

(vii) 10 Gaussian models, each with 500 entities, six features and partitioned into five Gaussian clusters with a mixture coefficient equal to 0.2.

(viii) Five Gaussian models, each with 500 entities, 15 features and partitioned into five Gaussian clusters with a mixture coefficient equal to 0.2.

(ix) Five Gaussian models, each with 1,000 entities, 25 features and partitioned into 12 Gaussian clusters with a mixture coefficient equal to 1/12.

(x) Five Gaussian models, each with 1,000 entities, 50 features partitioned into 12 Gaussian clusters with a mixture coefficient equal to 1/12.

The Gaussian models were generated with help from Netlab software and all have spherical clusters of variance 0.1. Their centres were independently generated from a Gaussian distribution N (0,1) of zero mean and unity variance.

One of the main purposes of most of the algorithms in this work is to discern the degree of importance of each feature. Unfortunately, this degree is not really known in the real datasets and is equal for each feature in the Gaussian models. With this in mind, we decided to add features containing uniformly distributed random noise to most of the datasets above. This noise is within domain, meaning that its maximum and minimum values are equal to the maximum and minimum values of the dataset. We have conducted a number of experiments with different quantities of noise features, as shown in Section 3.5.2.

Following, we present examples of the impact of the noise features in a few datasets. These were plotted on the plane of the first two principal components.

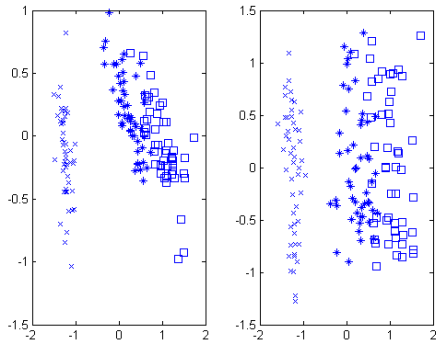

Figure 3.1: On the left, the iris dataset, on the right its version with two noise features, on the plane of the first two principal components, with the clusters shown using different shapes for entities

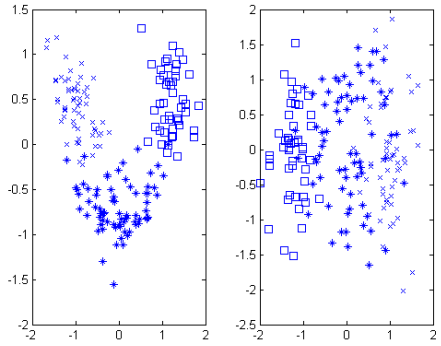

Figure 3.2: On the left, the wine dataset, on the right its version with 13 noise features, on the plane of the first two principal components, the clusters shown using different shapes for entities

The wine dataset was fairly well separated, but as Figure 3.2 shows, this ceased to be the case with the addition of 13 noise features.

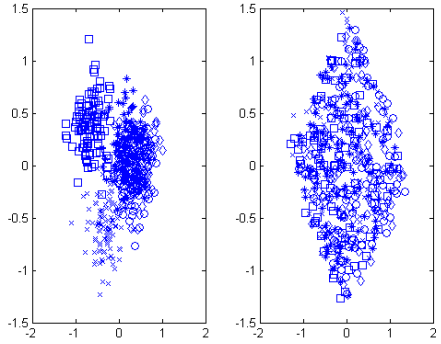

Figure 3.3: Illustration of one of the generated GM datasets with 500 entities, six features and five clusters on the left, and its version with two extra noise features on the right. Both are on the plane of their respective first two principal components.

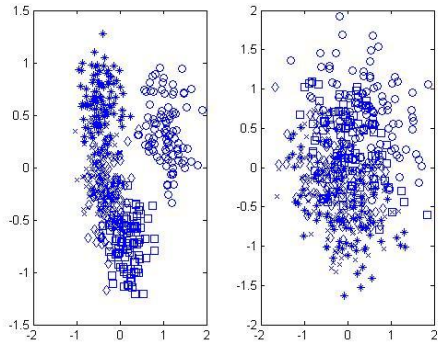

Figure 3.4: Illustration of one of the generated GM datasets with 500 entities, 15 features and five clusters on the left, and its version with 10 extra noise features on the right. Both are on the plane of their respective first two principal components.

Figures 3.3 and 3.4 show how noise affects the Gaussian mixtures. We have not plotted the datasets with 1,000 entities partitioned into 12 clusters, because even without noise these would be difficult to interpret as there would be 12 different shapes for the entities – one for each cluster.

All of the data, with or without random noise, has been standardised. The standardisation of numerical features follows Equation 3.13, below:

$$y_{iv} = \frac{x_{iv} - a_v}{0.5b_v}, i \in I, v \in V.$$

Equation 3.13: Standardisation of numerical features

in the above, a_v is the average value of feature v while b_v is its range, the difference between its maximum and minimum. In statistics, the standard deviation is conventionally used in the denominator of the Equation 3.13. However, empirical investigations demonstrated that in clustering, normalization by range leads to a better recovery of cluster structures Milligan and Cooper 1988, Steinley 2004). This phenomenon is explained in Mirkin (2005) as follows. If we standardise the data using the standard deviation we would bias data in favour of unimodal distributions, whereas

bimodal distributions should have a higher contribution to clustering (Mirkin, 2005:65).

We have standardised categorical features in a slightly different way to Huang et al. (2008): their method involves combining K-Means and K-Prototype in which centroids are represented by modal categorical values and averaged quantitative values when numerical. In our approach, we remain faithful to the original K-Means and follow a strategy close to the one described by Mirkin (2005:64). In this strategy, we represent a categorical feature with a quantitative binary feature, a dummy that is assigned one to each entity which falls in the category and zero if it does not. If a categorical feature has four categories, for example, then the final data matrix will have four dummy binary features instead.

The dummy features are then standardised quantitatively by subtracting its grand mean; in other words, the category's proportion. In our approach, the centroids are then represented by a categorical feature's proportion rather than its modal value.

Of course, we acknowledge that different standardisations may generate different results. To avoid any such issue for the Australian credit approval and heart disease datasets, we will provide here the results we obtained with WK-Means and the results obtained by Huang et al. (2008).

Another issue with the Australian credit approval dataset is that we have used the only dataset we could find: the one with 690 instances. In this dataset, all missing values have been substituted by their feature's average or modal value, for numerical and categorical features respectively, while Huang et al. (2008) use only 666 instances, the ones with no missing values.

3.5 Experimental results and comparisons

We have divided the results into two sections: 3.5.1, where we present the results of the algorithms with the original datasets in which no random noise features are included, and 3.5.2, in which the datasets do contain random noise features; the exact quantity of such features is stated before the results and in the tables' caption showing the experiments' results.

Both sections present the results of the real and the synthetic datasets, and where applicable the values of β exponent adjusted at the best accuracy of cluster recovery, as in Huang et al.'s (2008) experiments. We also further extend the results by reporting the maximum, average accuracy as well as the standard deviation, as these may help to further evaluate how the algorithms would perform with unlabelled data. The average accuracy of the six algorithms (K-Means, WK-Means, MWK-Means, iK-Means, iWK-Means and iMWK-Means) gives a general performance, which is what one could expect from the algorithm in scenarios where labels cannot be used to measure their performance because they are not present.

In order to find the optimalβ, we have run experiments across the interval of [1.0, 5.0] in steps of 0.1.

3.5.1 Experiments on original datasets

All of the experiments in this section were conducted with the original datasets, and no extra noise features were added.

(i) The iris dataset

We will firstly deal with the real-world datasets, starting with the iris dataset.

Algorithm	Exponent β at		% accuracy		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	84.0	12.3	89.3
WK-Means	2.0	1.8	87.1	13.8	96.0
MWK-Means	1.2	1.2	93.3	8.3	96.7
iK-Means	2.0	0.0	88.7		
iWK-Means	2.0	1.1	96.7		
iMWK-Means at different	2.0	2.0	94.7		
β	3.0	3.0	90.0		
	1.2	1.2	96.7		

Table 3.1: Accuracies achieved at different versions of K-Means clustering at the original iris dataset

In the above table, both iWK-Means and iMWK-Means reached the optimal accuracy of 96.7% at rather closeβ's of 1.1 and 1.2, respectively, meaning that only five instances were misclassified. In terms of MWK-Means, one of the algorithms which do not feature the anomalous pattern initialisation of centroids, the same accuracy appears in the maximum column but was in fact achieved once in a hundred trials. MWK-Means had average and maximal accuracies higher than WK-Means and K-Means.

Our result of 96.7%, obtained with iMWK-Means, compares favourably with the accuracy achieved by other clustering algorithms, reported in the recent literature,

as we show in Table 3.2. It is interesting to note that without a feature weighting scheme, even supervised and semi-supervised classification algorithms (which normally tend to have better results) such as SVM (Koggalage and Halgamuge, 2004) and SS-NMF (Chen et al.,2008) could not achieve such a high result.

	% accuracy	Comments
FCM + MSD	93.3	Combined fuzzy c-means and multidimensional scaling (Zhong et al., 2009)
SWFCM	91.3	A weighted version of fuzzy c-means (Fan et al., 2009)
TILDE	94.0	Top-down induction of logical decision trees (Blockeel et al.,1998)
CEFS	92.56 ±2.96	Clustering ensemble based method (Hong, 2008)
SS-NMF	92.7	Semi-supervised non-negative matrix factorisation (Chen et al., 2008)
Fast SVM	94.7	Supervised fast support vector machine technique separately classifying each of the classes (Koggalage and Halgamuge, 2004)
iWMK-Means	96.7	At $\beta = 1.2$

Table 3.2: Comparing clustering results achieved by various algorithms in the original iris dataset

It is a good idea to analyse how feature weights have behaved in iMWK-Means, as this is one of the key elements of our algorithm. These weights are shown per feature and per cluster in Table 3.3.

			Features			
			1	2	3	4
Initial anomalous clusters	Clusters	1	0.0022	0.0182	0.9339	0.0458
		2	0.0000	0.0000	0.9913	0.0086
		3	0.0000	0.0000	1.0000	0.0000
Final	Clusters	1	0.0228	0.1490	0.5944	0.2338
		2	0.0508	0.0036	0.5898	0.3558
		3	0.0233	0.0386	0.4662	0.4719

Table 3.3: iMWK-Means cluster-specific feature weights in the iris dataset at β=1.2

The above table shows that Features 3 and 4 (petal length and petal width) have considerably higher weights than the other features. This seems to be correct, as the literature using such datasets tends to show these features, or their product, as being the informative ones. It is also interesting to note how the values of the weights of such features differ according to their cluster.

(ii) The wine dataset

Table 3.4 presents the results of all six algorithms in the wine dataset. In contrast to the iris dataset, the feature weights scheme does not seem to improve the accuracy in this rather well-separated dataset. In this dataset, the generic K-Means seems to have the best performance.

Algorithm	Exponent at		% accuracy		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	95.3	0.4	96.6
WK-Means	2.0	4.4	93.9	0.8	94.9
MWK-Means	2.3	2.3	92.6	1.3	96.1
iK-Means	2.0	0.0	94.9		
iWK-Means	2.0	1.2	94.9		
iMWK-Means at different	1.2	1.2	94.9		
β	2.0	2.0	92.1		
	3.0	3.0	93.8		

Table 3.4: Accuracy levels achieved by the six versions of K-Means in the original wine dataset

The next table shows the optimum results of other algorithms. They appear to be comparable, as the maximum they could achieve is equivalent to the best run of the generic K-Means: 96.6%. Our algorithm, iMWK-Means was competitive at 94.9%.

	Accuracy, %	Comments
CEFS+PBIL	87.07±0.21	Ensemble based method + incremental algorithm (Hong et al., 2008)
SGKK-Means	91.60±2.12	Soft geodesic kernel K-Means (Kim et al., 2007)
NK-Means	95.8	A neighbourhood-based initialisation for K-Means (Cao et al., 2009)
SWFCM	96.6%	A weighted version of fuzzy c-means (Fan et al., 2001)

Table 3.5: Accuracy achieved by other clustering algorithms in the original wine dataset

(iii) The Pima Indians diabetes dataset

The results shown in the next table demonstrate that the Minkowski metric-based algorithms do outperform their counterparts; included is the average and maximum accuracy obtained with the WK-Means algorithm. The overall accuracy obtained by the six algorithms is not high, however, and can be surpassed by methods that readjust clusters to reduce the number of misclassifications, as presented by Zeidat and Eick (2004).

Algorithm	Exponent at		% accuracy		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	66.7	0.55	66.8
WK-Means	2.0	4.5	64.5	2.98	66.3
MWK-Means	3.9	3.9	68.2	2.85	71.3
iK-Means	2.0	0.0	66.8		
iWK-Means	1.8	1.8	64.7		
iMWK-Means	4.9	4.9	69.4		

Table 3.6: Accuracy levels achieved by the six versions of K-Means in the original Pima Indians diabetes dataset

In Table 3.7, we show a comparison with other algorithms under the same dataset. Zeidat and Eick (2004) achieve the best result after 50 random initialisations of their algorithm.

	% accuracy,	Comments
PAM	65.6	Partitioning around medoids (Kaufman and Rousseeuw, 1990), in Zeidat and Eick (2004)
SPAM	77.2	A supervised clustering algorithm that uses a PAM-based strategy to minimise the number of wrongly assigned objects (Zeidat and Eick, 2004)
SRIDHCR	79.5	A supervised clustering algorithm that uses Insertion/deletion hill climbing to minimise the number of wrongly assigned objects (Zeidat and Eick, 2004)
iMWK-Means	69.4	Proposed Minkowski metric-based

Table 3.7: Accuracy achieved by other clustering algorithms on the original Pima Indians' diabetes dataset

(iv) The hepatitis dataset

We will now analyse the accuracies of the six algorithms in the hepatitis dataset. The greatest level of accuracy, 84.5%, was obtained by iMWK-Means; this is higher than the maximum accuracy obtained by any other algorithm in Table 3.8.

Algorithm	Exponent at		% accuracy		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	71.5	1.36	72.3
WK-Means	2.0	1.0	78.7	0.13	80.0
MWK-Means	1.0	1.0	79.0	0.84	80.0
iK-Means	2.0	0.0	72.3		
iWK-Means	1.0	1.0	78.7		
iMWK-Means	2.3	2.3	84.5		

Table 3.8: Accuracy levels achieved by the six versions of K-Means in the original hepatitis dataset

The results obtained by iMWK-Means compare favourably to the results of other algorithms on the same data recently published by Belacel et al. (2007), and Zainuddin and Lye (2010). The algorithms in both publications have a different objective to ours. Here we are interested in clusters, while they both extend the training much further, producing classifiers. A comparison of the results can be seen in Table 3.9.

	% accuracy	Comments
RBFC +HKM	79.3 (0.14)	Radial basis functions classification network trained by using Harmony K-Means' results (Zainuddin and Lye, 2010).
PROAFTN with RVNS	85.8	Prototype-based fuzzy techniques multi-criterion decision method involving several manually set parameters (Belacel et al., 2007).
MSDD algorithm	80.8	Multi-stream dependency detection (MSDD) algorithm (Belacel et al., 2007).
iMWK-Means	84.5	Proposed Minkowski metric-based.

Table 3.9: Accuracy achieved by other clustering algorithms in the original hepatitis dataset

(v) Australian credit-approval dataset

We have also compared our results with those provided by Weighted K-Means presented by Huang et al. (2008) in two datasets from the same repository: Australian credit approval and heart disease. As discussed in Section 3.4, the standardisation we use is not exactly the same as Huang's, so we also present our results for WK-Means.

	Exponent β at		% accuracy		
	Distance	Weight	Mean	Std	Max
WK-Means combined with prototype (Huang et al., 2008)	2.0	9	71.93	*	85
WK-Means	2.0	4.9	71.82	13.71	86.52
iWK-Means	2.0	1.8	85.51	-	-
iMWK-Means	1.8	1.8	86.09	-	-

Table 3.10: Results of different weighted partitioning methods in the original Australian credit-approval dataset. * The standard deviation was not reported in Huang et al. (2008)

(vi) The heart disease dataset

Table 3.11 shows similar experiments under the same conditions for the heart disease dataset.

	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means/prototype (Huang et al., 2008)	2.0	9.0	73.55	*	85
WK-Means	2.0	4.2	78.50	6.22	82.59
iWK-Means	2.0	3.8	80.37	-	-
iMWK-Means	2.7	2.7	84.07	-	-

Table 3.11: Results of different weighted partitioning methods in the original heart disease dataset. * Huang et al. (2008) did not report the standard deviation

From the two tables above, for datasets (v) and (vi), we can conclude that iMWK-Means' results are indeed superior to those of WK-Means, and competitive

with the very best results achieved over one hundred runs of WK-Means/prototype method as shown by Huang et al. (2008).

(vii) 10 Gaussian mixtures

We will now analyse the accuracy of the six algorithms with synthetic data. In this set of experiments we use ten Gaussian models with 500 instances, six features and partitioned into five clusters.

	Exponent β at		% accuracy		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	79.8	9.1	89.4
WK-Means	2.0	4.2	78.2	10.6	91.8
MWK-Means	2.5	2.5	77.5	10.7	91.2
iK-Means	2.0	0.0	82.6	7.7	89.4
iWK-Means	2.0	4.5	81.0	9.9	89.8
iWMK-Means at different β	2.4	2.4	81.3	9.2	90.0
	2.0	2.0	75.4	9.9	88.8
	3.0	3.0	78.3	12.5	88.2

Table 3.12: Average accuracy results for K-Means' versions under consideration in the 10 original GM datasets. The best values of exponent β for WMK-Means and iWMK-Means versions are given for the single best overall β for the 10 GMs.

In this experiment, the within-cluster feature weights were considerably similar. This is not surprising because of the way the data is generated where all of the clusters and features have the same degree of relevance. The results in the Table 3.12 for

maximum and average accuracy are rather similar between the algorithms; this is because the Gaussian models do not confer any special advantage to weighted or anomalous clusters. This contrasts with the results seen in the real datasets, where intelligent versions of K-Means (the ones based on anomalous patterns) tend to outperform the conventional multiple-run schemes. We imagine this would suggest that the anomalous cluster mode is in general more suitable for real data than to those generated as the synthetic data we have.

3.5.2 Experiments on modified datasets

In this section, we present the results of experiments with modified versions of the datasets. This modification includes features made of random noise. We aim to show that Minkowski and intelligent versions of weighted K-Means do remain steady, or perform better against these added noise features than Weighted K-Means (Chan et al., 2004; Huang et al., 2005, 2008). This would be a compelling advantage over algorithms that are sensitive to data with noise features, such as the original K-Means and intelligent K-Means. We will firstly analyse the results from the iris dataset.

(1) The iris dataset with noise features

We have run experiments with two and four extra-uniformly distributed random noise features, which represent an increase of 50% and 100% in the number of features, respectively. The results can be found in Table 3.13, in the first and second line of each row, respectively.

	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	67.1	6.4	76.7
			66.7	7.0	80.0
WK-Means	2.0	1.2	85.5	15.8	96.0
			88.7	12.6	96.0
MWK-Means	1.2	1.2	88.2	16.9	96.0
			90.0	12.8	96.0
iK-Means	2.0	0.0	68.7		
			69.3		
iWK-Means	2.0	1.1	96.0		
			96.0		
iMWK-Means at different β	2.0	2.0	90.7		
			91.3		
	3.0	3.0	82.7		
			87.3		
	1.1	1.1	96.0		
			96.0		

Table 3.13: Accuracy levels achieved at different versions of K-Means clustering in the modified iris dataset; in each row, the first line represents the accuracy at two noise features, and the second line the same at four noise features

It is interesting to note that the optimal β for both datasets, with two and four extra noise features for the WK-Means algorithm is the same, although the actual results differ slightly. The same happens with MWK-Means at β 1.2 and iMWK-Means at β 1.1.

Another interesting point is that the noise features at both versions of the iris

dataset, with two and four extra noise features, had a weight close to zero in each cluster. MWK-Means produces slightly better results on average than WK-Means. IMWK-Means and IWK-Means are able to output the maximum results obtained by MWK-Means and WK-Means with considerably less computations owing to their anomalous pattern-based initialisation. Next, we present the results for the wine dataset.

(ii) The wine dataset with noise features

As before, the first and second lines of each row present the results for the dataset with an increase of 54% (the closest we can get to 50% in this case) and 100% in the number of features, made of uniformly distributed random noise. This represents seven and 13 extra noise features, respectively.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	93.0	6.5	96.6
			87.5	11.0	93.3
WK-Means	2.0	2.7	91.9	7.1	95.5
		2.6	89.4	10.6	94.9
MWK-Means	1.6	1.6	92.2	6.8	95.5
	1.4	1.4	88.3	10.6	94.4
iK-Means	2.0	0.0	94.4		
			93.3		
iWK-Means	2.0	1.9	94.9		
		3.0	93.8		
iMWK-Means	2.0	2.0	93.8		
			93.8		
	3.0	3.0	87.6		
			57.3		
	2.2	2.2	95.5*		
	1.1	1.1	94.9**		

Table 3.14: Accuracy of the K-Means' versions in the wine dataset, with seven and 13 noise features, shown on the first and second lines, respectively. *Optimal for the dataset with seven extra noise features. ** Optimal for the dataset with 13 extra noise features

The summary weight of the noise features on the anomalous cluster step in the wine dataset with 13 random noise features is less than 1% of the total. Although this increases to about 10% of the total at the final output, we found through experimentation that stopping the weights' update in the second part of the algorithm

considerably reduced the accuracy of the iMWK-Means. Again, iMWK-Means outperforms WK-Means on average, by outputting the maximum result obtained by the latter.

(iii) The Pima Indians diabetes dataset

We added four and eight features (first and second rows in Table 3.15 respectively) made of uniformly random noise to the original dataset. Again, this is an increase of 50% and 100% in the number of features.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	52.3	1.1	53.4
			51.8	1.2	53.6
WK-Means	2.0	1.9	65.5	3.7	68.4
		1.7	66.3	3.2	69.3
MWK-Means	2.0	2.0	63.6	5.4	68.4
	1.8	1.8	66.4	3.5	69.8
iK-Means	2.0	0.0	52.9		
			50.5		
iWK-Means	2.0	1.5	66.8		
		2.0	69.0		
iMWK-Means	2.0	2.0	66.3		
			66.1		
	3.0	3.0	67.1		
			66.7		
	4.1	4.1	67.2*		
	2.6	2.6	67.2**		

Table 3.15: Accuracy of the K-Means' versions at the Pima Indians diabetes dataset with four and eight noise features, shown on the top and bottom lines, respectively. *Optimal for the dataset with four extra noise features. ** Optimal for the dataset with eight extra noise features.

In the above experiment, the algorithms performed in a similar manner, with the exception of K-Means and iK-Means, suggesting that feature weighting increases the accuracy. There was little or no difference in the results when applying the algorithms to the dataset with four or eight noise features.

(iv) The hepatitis dataset

Originally, this dataset had 19 features. From it, we generated two datasets with 10 and 20 extra features made of uniformly distributed noise. The results for each dataset are presented in Table 3.16, in the first and second line of each row, respectively.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
K-Means	2.0	0.0	71.2	1.9	72.9
			70.0	1.7	71.0
WK-Means	2.0	1.0	78.7	0.0	78.7
		1.5	79.4	6.4	85.2
MWK-Means	1.0	1.0	78.8	0.1	80.0
	1.9	1.9	80.2	3.6	85.2
iK-Means	2.0	0.0	72.3		
			71.0		
iWK-Means	2.0	1.0	78.7		
		2.3	84.5		
iMWK-Means	2.0	2.0	63.9		
			64.5		
	3.0	3.0	63.2		
			60.6		
	4.5	4.5	83.3*		
	4.5	4.5	79.3**		

Table 3.16: Accuracy of the K-Means' versions in the hepatitis dataset with 10 and 20 noise features, shown on the first and second lines of each row, respectively. *Optimal for the dataset with four extra noise features. ** Optimal for the dataset with eight extra noise features.

In the above, the optimal β for WK-Means, MWK-Means, iWK-Means for the dataset with 10 extra features was the same: one. The iMWK-Means algorithm produced the best results with nearly 5% difference from WK-Means, outperforming its maximum. The cluster recovery obtained with iMWK-Means in the dataset with 20 extra noise features was very similar to the average obtained with WK-Means.

We observed that increasing the number of noise features has resulted in slightly better results for the Iris, Pima Indians diabetes and Hepatities datasets (Tables 3.13, 3.15 and 3.16, respectively). There can be two possible explanations:

(1) Although this small increase did appear in the mentioned datasets, it did not appear in all other experiments, suggesting it may not have statistical significance.

(2) In datasets with many noise features, each distance is affected similarly. This means the minimum is reached with the correlation between distance matrices, which are more correlated to the original distances.

Because of the small number of occurrences of this unexpected increase, we intend to address this in future research.

We did not conduct experiment with increasing the number of features in the Australian credit approval (v) nor the heart disease (vi) datasets with uniformly distributed random noise, because the objective of these two datasets was to enable a comparison with the work of Huang et al. (2008). Following this, we present further comparative results applying all six algorithms to the Gaussian mixtures.

(vii) 10 Gaussian models

In our first experiments, two random noise features were added to 10 Gaussian models. Each of these contained 500 entities and six original features and was partitioned into

five clusters. Table 3.17 summarises the results obtained in the 10 datasets; the results per dataset can be found in Appendix A, Tables A.1 to A.4.

Algorithm	Exponent β at Weight		Accuracy, %			
	Mean	Std	Mean	Std of Mean	Max	Std of Max
K-Means	-	-	38.5	5.29	47.6	6.44
WK-Means	1.50	0.08	56.6	5.26	78.6	9.45
MWK-Means	1.54	0.07	60.3	6.37	80.5	8.70
iK-Means	-	-	-	-	37.7	6.86
iWK-Means	2.14	1.00	-	-	66.1	11.58
iMWK-Means	1.79	0.48	-	-	70.3	11.91

Table 3.17: Accuracy levels at the considered versions of K-Means for the 10 GM datasets with 500 entities, six features and five clusters with an additional two random-noise features.

Table 3.17 shows the results of applying the six K-Means based algorithms to the noisy versions of a Gaussian model, and consistently shows that:

i. The accuracy of the results is considerably less than in the experiments with no added noise features (see Table 3.8). This is reflected in the fact that the weights for the original features were two to three times higher than that of the two noise features, being around 0.13-0.16 and 0.05-0.06, respectively. This also contrasts with the near-zero weights of noise found in the experiments conducted with real datasets. However, MWK-Means achieved the highest maximum, with 80.5% accuracy.

ii. On average, the maximum accuracy of Minkowski Weighted K-Means is greater than that of WK-Means: of 80.5% and 78.6%, respectively.

iii. The intelligent version of K-Means presents a rather modest accuracy, probably because of the previously cited lack of structure in the synthetic datasets.

iv. The ‘intelligent’ weighted algorithms tend to outperform the average obtained by their counterparts over 100 runs, with iMWK-Means achieving the best performance. This shows that when clustering unlabelled data, the intelligent Minkowski metric K-Means remains a viable option, even when the data structure is vague.

On average, these experiments demonstrate that the versions of WK-Means that use the Minkowski metric are quite competitive, and frequently superior to, the Euclidean metric alternatives in both the original datasets and their noise versions.

In order to further validate our proposed algorithms and to see how they perform in larger feature spaces we have conducted further experiments, increasing the amount of features and entities.

(viii) Five Gaussian models, each with 500 entities and 15 features

Our second set of experiments with synthetics used five Gaussian models of 500 entities and 15 features, partitioned into five spherical clusters. To these datasets we added 10 features made of within-domain uniformly distributed random noise. The summary of results for all of the tested algorithms can be found below, and the results per dataset in Appendix A:

Algorithm	Exponent β at Weight		Accuracy, %			
	Mean	Std	Mean	Std of Mean	Max	Std of Max
K-Means	-		46.7	9.57	62.8	8.86
WK-Means	3.44	1.55	72.4	9.97	89.8	9.71
MWK-Means	1.40	0.07	78.8	6.58	93.4	4.24
iK-Means	-	-	-	-	55.8	16.29
iWK-Means	3.18	1.44	-	-	80.8	9.64
iMWK-Means	1.60	0.27	-	-	88.8	9.25

Table 3.18: Accuracy levels at the considered versions of K-Means for the five GM datasets with 500 entities, 15 features and five clusters with additional 10 random noise features

Table 3.18 presents a rather similar pattern to Table 3.17. Algorithms using the Minkowski metric tend to outperform their counterparts, and those using the 'intelligent' initialisation provided by the use of anomalous patterns also tend to, on average, outperform those based on multiple runs. Unsurprisingly, K-Means is the algorithm that suffers the most and performs the poorest. Table 3.18 suggests that weighted algorithms based on the Euclidean metric may be less resistant to noise than those based on the Minkowski metric.

The results in Table 3.18 reinforce the point that MWK-Means has a considerably smaller standard deviation for β.

(ix) Five Gaussian models with 1,000 entities and 25 features

In our next set of experiments, yet with synthetic data we generate another five Gaussian models. These were originally made of 1,000 entities and 25 features

partitioned into 12 different clusters. To these datasets, we added a further 25 features made of within-domain uniformly random noise. Table 3.19 summarises the results; the results per dataset can be found in Appendix A.

Algorithm	Exponent β at Weight		Accuracy, %			
	Mean	Std	Mean	Std of Mean	Max	Std of Max
K-Means	-	-	20.7	1.84	28.6	3.79
WK-Means	4.76	0.23	52.0	5.36	71.4	6.66
MWK-Means	1.32	0.04	68.4	3.15	84.1	1.47
iK-Means	-	-	-	-	21.9	5.82
iWK-Means	2.88	1.36	-	-	39.3	14.47
iMWK-Means	1.48	0.18	-	-	54.9	3.27

Table 3.19: Accuracy levels at the considered versions of K-Means for the five GM datasets with 1,000 entities, 25 features and 12 clusters with an additional 25 random noise features

There results in Table 3.19 allow us to conclude that in all five Gaussian models, the highest average accuracy was achieved by MWK-Means, which suggests this algorithm could be the most resistant to an increase in the rate of noise features in relation to the overall number of features, as well as to an increase in the number of clusters which was increased from five in the previous synthetic data experiments to 12 in this one.

Another interesting point is that the optimal value for β continued to be considerably more stable while using MWK-Means than WK-Means. In all experiments with this algorithm, the results obtained with β 1.3 and 1.4 were either at the optimal value or had a negligible difference in relation to their optimal accuracy.

Although the 'intelligent' versions had a poor result, considerable β stability can be seen in iMWK-Means in relation to iWK-Means.

It also seems that after this increase of features there was a shift from the best algorithms being those using the 'intelligent' initialisation to those based on multiple runs.

(x) Five Gaussian models, each with 1,000 entities and 50 features

In this final set of experiments we generated five Gaussian Models with 1,000 entities and 50 features, partitioned it into 12 clusters and added 25 features containing within-domain uniformly random noise. Table 3.20 summarises the results obtained in all datasets; the results per dataset can be found in Appendix A.

Algorithm	Exponent β at Weight		Accuracy, %			
	Mean	Std	Mean	Std of Mean	Max	Std of Max
K-Means	-	-	92.8	1.07	99.9	0.04
WK-Means	4.18	0.80	80.4	1.08	93.1	0.62
MWK-Means	1.22	0.04	85.2	8.18	98.7	02.90
iK-Means	-	-	-	-	88.5	8.15
iWK-Means	1.84	1.34	-	-	37.4	25.41
iMWK-Means	1.90	0.47	-	-	70.8	9.84

Table 3.20: Accuracy levels at the considered versions of K-Means for the five GM datasets with 1,000 entities, 50 features and 12 clusters with an additional 25 random noise features

Table 3.20 seems to suggest that a change in pattern is complete. The weighted algorithms based on the Minkowski metric still outperform their Euclidean-metric counterparts; however, it is K-Means that presents the best results, in contrast to those using the 'intelligent' initialisation, which on this occasion has produced rather poor results.

A possible explanation for the above scenario may come from the effects of the higher dimension sizes explained in Aggarwal et al. (2000). With the distances being relatively small and similar to each other, the 'intelligent' initialisation can no longer discern 'anomalous' patterns. In the datasets we use here the relevant features prevail, being two thirds of the total amount of features; they also have similar weights: These facts may enable K-Means' good cluster recovery.

3.6 Time comparisons

We have conducted a number of experiments to consider the computational intensity of the various versions of K-Means dealt in this thesis.

The experiments calculate the average CPU time based on 100 runs of each multi-run algorithm (K-Means, WK-Means, MWK-Means) and a single run of the deterministic algorithms (iK-Means, iWK-Means and iMWK-Means). For those algorithms that need a β we have experimented (100 runs or one, depending if the given algorithm is multi-run or deterministic) in the range from 1.0 to 5.0, inclusive, in steps of 0.1, in line with all our previous experiments. Table 3.21 shows the average CPU time of each algorithm per run.

Dataset	Algorithm					
	K-Means	iK-Means	WK-Means	iWK-Means	MWK-Means	iMWK-Means
500x(6+2)	0.03	0.08	0.08	0.11	6.59	9.40
500x(15+10)	0.08	0.14	0.19	0.32	22.89	29.82
1000x(25+25)	2.28	2.07	1.05	1.65	119.50	208.23
1000x(50+25)	0.20	0.65	1.05	0.94	124.47	153.96
Iris (+2)	0.01	0.02	0.02	0.45	0.71	1.36
Iris (+4)	0.01	0.02	0.02	0.61	0.99	1.65
Wine (+7)	0.01	0.02	0.05	1.54	3.83	5.69
Wine (+13)	0.02	0.03	0.07	1.69	4.72	6.82
Pima Indians diabetes (+4)	0.02	0.05	0.04	2.13	17.65	19.57
Pima Indians diabetes (+8)	0.04	0.04	0.06	2.92	26.40	31.60
Hepatitis (+10)	0.02	0.06	0.11	3.80	2.30	3.02
Hepatitis (+20)	0.02	0.06	0.11	7.12	3.06	4.51

Table 3.21: The average amount of time in seconds for a single run for all algorithms under consideration. Note that unlike the others, "intelligent" algorithms (iK-Means, iWK-Means and iMWK-Means) need to be run only once to get an optimal result. The number of added noise features is in the parenthesis.

The amount of time taken by a given algorithm depends on different factors based on hardware and software. In terms of hardware, the above experiments were run on a computer with an Intel Core-2 CPU of 2.4 GHz, with 2GB of RAM. The programs were all written in the MatLab environment, version 2010 which ran under Windows 7.

The Table 3.21 provide us with a good indication of how the algorithms would perform in a real life situation. We can clearly see that both, the Anomalous pattern initialisation found in the 'intelligent' algorithms, and the feature weighting in Euclidean space do not add much in terms of CPU time when compared with K-Means.

The Minkowski's metric based algorithms did not have such a good performance, taking one hundred or more times longer to converge than K-Means. This happens because these algorithms need to find the Minkowski centres in each cluster at each iteration. The centre of gravity of a cluster in all other algorithms is given by the average, which is much faster to compute. However, in order to find a good clustering WK-Means was run a hundred times in our previous experiments, balancing the CPU time of both, with the difference that iMWK-Means found clusters closer to the given labels.

We find rather interesting to see that it is possible for WK-Means and iMWK-Means to take less time to converge than a hundred runs of K-Means in very noisy datasets, as it happened with the GM dataset of 1000 entities over 25 features with extra 25 features of noise.

3.7 Conclusion

In this chapter, we have presented two contributions. Firstly, we introduced the Minkowski metric criterion (Equation 3.6), which extends the effect of the exponent β from the weights adjustment as in the original Weighted K-Means (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008) to the distances between. This new criterion follows the original K-Means criterion in the sense that it stills attempts to minimise the summary of distances between entities and their clusters' centroids, but also makes the weights to be the feature rescaling coefficients. The Minkowski metric allowed us to overcome the drawback of Huang et al. (2005, 2008) lack of meaning of β.

We have empirically shown that the Minkowski metric does indeed improve Weighted K-Means' accuracy both at the original and noisy datasets with different

amounts of features and noise features. We believe this to be a further advance into the problem of clustering data that has its structure hidden by the presence of irrelevant, or noisy, features.

Our second contribution comes from utilising the anomalous cluster-based initialisation provided by iK-Means (Mirkin, 2005:93) to generate good starting centroids and weights. Both of these were originally started randomly in Weighted K-Means (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008). We have empirically shown that this initialisation tends to be beneficial in datasets with a modest to moderate number of features. We expected an improvement while experimenting with real-world datasets (i, ii, iii, iv, v and vi), because their structure is likely to be well aligned with the anomalous cluster mechanism (Mirkin, 2005: 90), unlike our synthetic datasets (vii, viii, ix and x), which have 'rounded' clusters. Even under such conditions, the 'intelligent' versions of the algorithms proved superior to their multiple run-based counterparts, on average across the synthetic datasets, as well. We believe this to be an important finding because in ordinary clustering scenarios, the value of the K-Means criterion itself (Equation 2.8) is not a reliable indicator as β is variable. Our experiments in high dimensions have shown a different pattern, however, in which the 'intelligent' initialisation ceased to provide good centroids, and this deserves further investigation.

In general, we have shown that given a good β, our proposed algorithms tend to provide a high level of accuracy, and the Minkowski metric-based algorithms tend to have a considerably more stable β.

The next issue is how to determine the optimal value for the exponent β. This is also an open question in the case of Weighted K-Means (Chan et al., 2004; Huang et

al., 2005; Huang et al., 2008). In an attempt to learn the optimal value for this exponent, in the next chapter we present experiments using semi-supervised learning.

Chapter 4: Learning the Minkowski exponent

4.1 Semi-supervised learning

In this chapter, we will attempt to ascertain whether the Minkowski metric β value can be learned from the data in a semi-supervised manner. In order to achieve this, we have conducted a series of experiments involving all of the datasets used for our previous computations (see Chapter 3, Section 3.4).

Unsupervised algorithms, such as K-Means and iK-Means, were designed to deal with scenarios in which there are no labelled samples that can be used to train a given algorithm. Such scenarios may occur because of the financial and time costs of labelling data samples. This subject was touched on in Chapter 2, Section 2.1.

In contrast, algorithms based on supervised learning tend to excel when there is a considerable amount of correctly labelled data available. The accuracy of the algorithms tends to conform to the quantity and quality of the labelled data. If there is insufficient labelled data, the algorithm may not be able to learn a good hypothesis and by consequence may fail in its generalisation. If too many samples are used in the learning process, the algorithm may suffer from overfitting, which can occur when the hypothesis relates to a possible noise, random or not, in the data, rather than to the relationship between the data and the labels; if some of the labels are incorrect or the labelled data does not closely represent the dataset, the algorithm may simply learn a suboptimal hypothesis, generating numerous misclassifications.

Traditionally, this issue has been clear-cut, and algorithms have used either a supervised approach for classification or an unsupervised one for clustering, depending on the availability of labelled data. Issues can arise if, for instance, there is a limited amount of labelled data. Although the quantity may not be enough to apply a

supervised learning approach, intuitively speaking it should improve the hypothesis.

A classical example of a semi-supervised learning scenario is the Internet. When clustering web pages into pre-defined clusters, it is easy to acquire a large amount of unlabelled data from the millions – or more of readily accessible pages, but the process of labelling a representative amount of such data is very time-consuming (Witten and Frank, 2005:337; Chapelle et al., 2006); in this case, it may be possible to label a very small amount of the data and use this labelled sample to improve accuracy.

Furthermore, human beings perform semi-supervised learning, as empirically shown by Zhu et al. (2007). There are, generally speaking, two views on this approach: (1) A supervised learning approach that also learns from unlabelled data; (2) An unsupervised learning approach that makes use of a small amount of labelled data.

In terms of the first view, self-learning (also known as self-training, self-labelling and decision-direct learning) is probably the earliest attempt to use unlabelled data in classification (Chapelle, 2006). Using this approach, algorithms start by training on the available labelled data, and label part of the unlabelled data according to the current hypothesis; then, the algorithm is retrained with all of the original and predicted labelled data (Scudder, 1965; Fralick, 1967; Agrawala, 1970).

As regards the second view, we have for instance constrained versions of K-means and distance training, as discussed in Section 4.2; these normally attempt to limit the clustering based on what is learnt from a small labelled data sample.

In this chapter, we will focus on this view of semi-supervised algorithms as being unsupervised algorithms that make use of a small amount of labelled data: this is because our algorithms, MWK-Means and iMWK-Means, are K-Means-based, and this is an unsupervised algorithm.

4.2 Semi-supervised learning in K-Means

Clustering is usually regarded as synonymous with labelling data using an unsupervised learning approach; however, it may be possible to acquire information, even if this is limited, regarding the problem domain. This information, if integrated into a clustering algorithm such as K-Means, may direct it to a better set of clusters.

There have been different approaches to this integration, but generally speaking they fall into two broad categories: constraint-based and metric-based (Bilenko et al., 2004).

(1) Constraint-based algorithms

Constraint-based algorithms are fed with pairwise rules, such as whether two entities should be clustered together or not. These rules are normally known as must-link and cannot-link (Wagstaff et al., 2001; Basu et al., 2002; Bilenko et al., 2004; Kumar and K. Kummamuru, 2007; Amorim, 2008): these rules may be pre-determined or constructed from a labelled dataset.

Wagstaff et al. (2001) present what is, to our knowledge, the first attempt to use semi-supervised learning in K-Means, introducing the constrained K-Means (COP-KMeans) algorithm (4.1).

1. *Initial setting*
   Similarly to K-Means, this method starts the centroids at random. It also requires as input two sets of pairwise rules, one for the pairs of entities that must be clustered together and one for the entities that should not be clustered together.

2. *Cluster update*
   This assigns each entity to the closest centroid, making sure it follows the must-link and cannot-link rules.

3. *Stop condition*
   As per the K-Means algorithm.

4. *Centroids update*
   Updates centroids to their centre of gravity. As the algorithm uses Euclidean distance, the centre of a cluster is its mean. Then, return to step 2.

Algorithm 4.1: COP-KMeans

In the cluster-update stage, Wagstaff et al. (2001) modify the K-Means clustering objective function to include the satisfaction of the given constraints. For instance, instead of clustering two entities which are close, the COP-KMeans algorithm will only cluster them if they are close and are not under a cannot-link rule. In terms of the must-link rule, if two entities are under this rule and one is assigned to a cluster S_k, the other is automatically assigned to it as well.

In a later publication, Wagstaff et al. (2006) acknowledge that although the average accuracy obtained with the constrained version of K-Means tends to exceed that of the unconstrained version, as intuitively expected, the use of constraints, even if they do not contain any noise and are of a fixed size, can produce results that are significantly worse than those elicited by not using constraints at all.

Furthermore, as expected, the COP-KMeans algorithm suffers from the poor initialisation inherited from K-Means itself. This issue has been addressed with the help of iK-Means in a previous publication (Amorim, 2008), where we introduce the constrained intelligent K-Means. This is shown in Algorithm 4.2.

1. *Initial setting*
   Specify the cluster-discarding threshold used to remove all anomalous clusters whose size is less than the threshold.
2. *Anomalous pattern*
   Apply the anomalous pattern algorithm: Algorithm 2.5. When performing the cluster update (step 3), assign entities to the closer centroid if it does not break the cannot-link rules. Cluster together entities under a must-link rule regardless of distance.
3. *Stop Condition*
   As per the intelligent K-Means algorithm.
4. *Removal of small clusters*
   Remove all clusters that have not reached a size equal to the cluster-discarding threshold.
5. *K-Means*
   Run COP-KMeans using the found centroids.

Algorithm 4.2: Constrained Intelligent K-Means

The above algorithm shows how easy it would be to integrate an anomalous pattern-based algorithm, such as iMWK-Means, with COP-KMeans.

Although the constraint-based category is seen as the most popular, it has two main drawbacks, as pointed out by Kumar and Kummamuru (2007):

(i) The entities in the cannot-link constraints may actually lie in the wrong clusters yet still satisfy the cannot-link constraints;

(ii) When the pairwise feedback is generated from the labelled part of the training set, the must-link constraints would mislead the clustering algorithm if the entities in the cluster belonged to two different clusters of the same class.

The first drawback could easily affect algorithms like COP-KMeans and CiK-Means. Kumar and Kummamuru (2007) suggest a partial solution in which the pairwise rules do not assume that entities belong to a cluster but use instead relative comparisons which assume only the entities' proximity.

(2) Metric-based algorithms

Metric-based algorithms, on the other hand, aim to train the metric of the algorithm to satisfy the given labels or constraints (Bilenko et al., 2004); in most cases, this happens by a weight adjustment.

In these algorithms, there is normally a clear separation between the metric learning which does not use unlabelled data and the clustering process per se. Although not as popular as the constraint-based type, there are a number of algorithms in this category. For instance, Kleinet et al. (2002) present a method which trains the Euclidean distance by using a shortest-path algorithm. Xing et al. (2003) have also produced work with the Mahalanobis distance, and Bar-Hillel et al. (2003) have trained the distance using a convex optimisation.

Algorithms that follow the approaches in both categories can be found in the work of Bilenko et al. (2004), and Kumar and Kummamuru (2007).

However, we will pursue a different approach here; rather than seeking the optimisation of a clustering by using a small amount of labelled data, as the above algorithms, we seek to learn the exponent β of iMWK-Means. This is an algorithm we have shown to be competitive or superior to WK-Means in different scenarios, in Chapter 3. Of course, the same processes we apply here can be used to learn the same exponent for MWK-Means.

4.3 Semi-supervised algorithms for β

We have carried out experiments with both WK-Means and iMWK-Means in order to reach a comparison. In general, in these experiments we apply one of the two algorithms to a dataset subset containing only a small proportion of labelled entities, 20% of the dataset. The aim is to find the a good value for β for the whole dataset, for this we find the optimal β in the subset, which is the one with the highest average accuracy within this small labelled dataset, and then apply the given algorithm to the whole dataset with the found β.

Semi-supervised algorithms tend to combine a limited amount of labelled data with a considerably larger amount of unlabelled data; in our experiments with WK-Means and iMWK-Means, we follow the same framework. In the experiments with both algorithms, the β is learnt from the limited amount of labelled data and the weights, from the unlabelled data, as before. Another reason to use two different datasets for learning the feature weights and the exponent β is to avoid feature-selection bias, which may occur when feature selection and classification learning take place in the same dataset (Liu and Modota, 2008).

To analyse the results we compute the mean, the mode and the proportion of the

runs in which the local optimal β is equal to the mode. We show the average accuracy together with its standard deviation and maximum, although this is of secondary importance here.

4.3.1 WK-Means

We begin by describing how our algorithm works. The WK-Means algorithm (Chan et al., 2004; Huang et al., 2005; Huang et al., 2008) was discussed in Section 3.2.1, and its steps can be found in Algorithm 3.1.

Algorithm 4.3 was used for the experiments in this section.

Run the following 100 times:

1. *Select the learning data*
   Randomly choose 20% of the data plus related labels.
2. *Parameter search*
   Run WK-Means, Algorithm 3.1, 100 times for each β ranging from 1.0 to 5.0, in steps of 0.1.
3. *Evaluation*
   Set the value of β^* to the β with highest average accuracy.
4. *WK-Means*
   Run WK-Means with β^* in the whole dataset 100 times.

Algorithm 4.3: Semi-supervised WK-Means experiments

In the above algorithm, arguably the most evident issue is its required computational effort. Our range of β is [1, 5], in steps of 0.1; thereby, WK-Means is run in each of the 100 loops in step 2 alone 4,100 times. We could have reduced the number of loops, but this would not seem fair or in accordance with our experiments in Chapter 3.

4.3.2 iMWK-Means

The iMWK-Means algorithm was introduced in Chapter 3, Section 3.3.3. We have used Algorithm 4.4 in the experiments this section.

Run the following 100 times:

1. *Select the learning data*
   Randomly choose 20% of the data plus related labels.
2. *Parameter search*
   Run a series of experiments with β ranging from 1.0 to 5.0, in steps of 0.1.
3. *Evaluation*
   Set the value of β^* to the β with the highest accuracy.
4. *iMWK-Means*
   Run iMWK-Means as discussed in Section 3.3.3 with β^* in the whole dataset.

Algorithm 4.4: Semi-supervised iMWK-Means experiments

In terms of computational effort, the above algorithm is visibly cheaper than 4.3 (WK) and, in consequence, faster. This happens because algorithms like iMWK-Means

and iK-Means, which use the anomalous pattern initialisation we described in Chapter 2, Section 2.4, are deterministic; therefore, for a given input the algorithm will always have the same output, so it will not be necessary to run it a number of times.

4.4 Experimental results and comparisons

We perform experiments with Algorithm 4.3 and Algorithm 4.4, in all datasets described in Chapter 3, Section 3.4.

4.4.1 Real-world datasets

We begin by applying the algorithm 4.3, which uses WK-Means, to the original real-world datasets (i to vi), with no added noise.

Dataset	Exponent β at weight			% accuracy		
	Mean	Modal	Optimal	Mean	Std of Mean	Max
Iris	2.98	2.2(8%)	1.8	83.9	2.55	90.0
Wine	3.85	4.5(9%)	4.4	92.5	0.54	93.6
Pima Indians	3.91	4.2(10%)	4.5	62.9	0.87	64.3
Hepatitis	1.38	1.0(83%)	1.0	77.5	2.7	78.8
Australian credit-card approval	4.41	4.5(11%)	4.9	68.8	2.8	73.0
Heart disease	4.00	4.7(10%)	4.2	76.6	2.0	79.4

Table 4.1: Semi-supervised estimation of β in WK-Means on original datasets (i to vi)

The results of the same datasets, but using the iMWK-Means algorithm, can be found in Table 4.2.

Dataset	Exponent β at weight			% accuracy		
	Mean	Modal	Optimal	Mean	Std of Mean	Max
Iris	1.44	1.0(31%)	1.2	93.9	1.88	96.7
Wine	1.74	1.2(22%)	1.2	93.1	1.35	94.9
Pima indians	3.47	3.1 (7%)	4.9	64.5	6.16	69.4
Hepatitis	1.57	1.0 (48%)	2.3	72.9	7.09	84.5
Australian credit-card approval	2.56	1.7 (9%)	1.8	67.0	12.76	86.1
Heart disease	2.25	1.4 (11%)	2.7	76.1	9.44	84.1

Table 4.2: Semi-supervised estimation of β in iMWK-Means on original datasets (i to vi)

In the above tables, although the modal β seems to be closer to the optimal in WK-Means than iMWK-Means, possibly because of the higher amount of runs necessary, it is the latter that has the best maximum accuracies.

Experiments were conducted on derivate versions of the original datasets, with an increase of 50% and 100% in the number of features with uniformly distributed, within-domain random noise. The exact number of features added to each dataset is shown after their name in parentheses. Table 4.3 shows the results for the WK-Means algorithm.

Dataset	Exponent β at weight			% accuracy		
	Mean	Modal	Optimal	Mean	Std of Mean	Max
Iris (2)	2.77	2.1(6%)	1.2	74.2	3.48	83.9
Iris (4)	2.72	1.0(8%)	1.2	75.2	5.68	90.5
Wine (7)	3.02	4.7(6%)	2.7	89.3	2.67	92.5
Wine (13)	1.73	1.2(32%)	2.6	77.5	4.57	88.0
Pima Indians (4)	1.47	1.5(32%)	1.9	63.8	1.17	65.2
Pima Indians (8)	1.49	1.5(32%)	1.7	65.6	1.43	66.5
Hepatitis (10)	1.2	1.0(93%)	1.0	78.4	1.45	78.8
Hepatitis (20)	1.26	1.0(89%)	1.5	78.6	0.5	78.8

Table 4.3: Semi-supervised estimation of β in WK-Means on modified datasets (i to iv)

The results for the same datasets using iMWK-Means can be found in Table 4.4.

Dataset	Exponent β at weight			% accuracy		
	Mean	Modal	Optimal	Mean	Std of Mean	Max
Iris (2)	1.28	1.1(31%)	1.1	94.5	2.47	96.0
Iris (4)	1.21	1.1(33%)	1.1	94.4	1.85	96.0
Wine (7)	1.49	1.3 (20%)	2.2	91.4	2.59	95.5
Wine (13)	1.32	1.1 (22%)	1.1	92.2	2.03	94.9
Pima indians (4)	2.42	2.0(8%)	4.1	61.7	7.37	67.2
Pima indians (8)	1.85	1.5(14%)	2.6	60.2	5.51	67.2
Hepatitis (10)	2.12	1.0(32%)	4.5	68.2	9.97	78.1
Hepatitis (20)	1.73	1.0(26%)	4.5	66.8	8.66	79.3

Table 4.4: Semi-supervised estimation of β in iMWK-Means on modified datasets (i to iv)

The pattern found in Tables 4.3 and 4.4 is rather similar to that in Tables 4.1 and 4.2, which present the datasets without noise. Again, it is difficult to see a constant relation between the mean and modal values of β in the datasets with their respective optimal values. The maximal accuracy obtained with the semi-supervised version of iMWK-Means was consistently competitive, or considerably superior, to the semi-supervised version of WK-Means.

4.4.2 Synthetic datasets

We have performed a number of experiments using the synthetic datasets described in Chapter 3, Section 3.4. To these were added features also consisting of uniformly random noise; the exact quantity per dataset is shown after their names.

Table 4.5 presents the results for the semi-supervised version of WK-Means. The results per dataset, together with the frequency of the modal β, also per dataset, can be found in Appendix B.

Dataset	Exponent β at weight					% accuracy			
	Mean	Std of Mean	Mean of Modal	Std of Modal	Mean Optimal	Mean	Std of Mean	Max	Std of Max
500x8 (2)	1.89	0.25	1.51	0.13	1.5	54.13	5.13	57.68	5.03
500x25 (10)	1.49	0.03	1.44	0.09	3.44	69.29	6.81	72.34	7.56
1000x50 (25)	1.38	0.02	13.4	0.05	4.76	46.51	3.9	49.19	3.7
1000x75 (25)	1.74	0.19	1.52	0.18	4.18	75.16	1.15	78.94	1.63

Table 4.5: Semi-supervised estimations of β in WK-Means on Gaussian models with noise, datasets vi to x

We present the results for the same datasets for the semi-supervised version of iMWK-Means in Table 4.6; again, the results per dataset and the frequency of the modal β, also per dataset, can be found in Appendix B.

Dataset	Exponent β at weight					% accuracy			
	Mean	Std of Mean	Mean of Modal	Std of Modal	Mean Optimal	Mean	Std of Mean	Max	Std of Max
500x8 (2)	1.81	0.12	1.45	0.07	1.79	55.8	9.63	70.14	11.96
500x25 (10)	1.61	0.06	1.4	0	1.6	74.6	10.98	89.8	10.21
1000x50 (25)	1.57	0.01	1.46	0.05	1.48	43.92	2.24	54.88	3.26
1000x75 (25)	1.92	0.12	1.38	0.08	1.90	51.0	3.37	70.76	9.83

Table 4.6: Semi-supervised estimation of β in iMWK-Means on Gaussian models with noise, datasets vi to x

We can observe a clear change in the pattern here. In Table 4.6, the mean of the exponent β is close to the mean of its optimal value per dataset in all cases. We can observe that the standard deviation of the mean of β seems to be inversely proportional to the rate of the noise features (the number of noise features divided by the total features), and that this standard deviation tends to decrease as the rate of the noise features increases.

4.5 Conclusion

As one can see in Tables 4.1 to 4.4, the experiment series with real-world data, with and without noise, even when a rather high proportion of the data is used for estimating β, it seems that there is not a straightforward relationship between the estimated values for β in the semi-supervised framework and those that were found to be the optimal

values in Chapter 3, Section 3.5, in both WK-Means and iMWK-Means. These experiments show that finding the best Minkowski metric value at datasets with unlabeled entities remains an open issue.

The experiments using Gaussian mixtures were considerably more successful. The results of these can be found in Tables 4.5 and 4.6; in the latter there is a clear similarity between the mean of the estimated β value and its optimal, also found in Chapter 3, Section 3.5. We were also able to observe a possible relationship between the standard deviation of the estimates and the rate of noise features, which were inversely proportional. Unfortunately, the accuracy of the semi-supervised version of iMWK-Means seems to deteriorate in high-dimensional datasets.

It is not possible to use a semi-supervised approach to attempt to find β if the labels are not available. In such case it is even more difficult to find a workable value for β. A Possible solution would be to generate a number of labelled Gaussian mixtures with different settings, but with the same number of features, entities and clusters as the dataset to be clustered. One could then apply iMWK-Means at different βs on the labelled data and use the β with the highest average accuracy on the dataset to be clustered. We leave the analysis this scenario to future research.

In Chapter 5, we will address scenarios in which the number of features is far greater than the number of entities.

Chapter 5: Reducing feature space in high-dimensional data

5.1 Introduction

In this chapter, we intend to analyse the behaviour of iMWK-Means and WK-Means when dealing with a dataset containing a large number of features, as this is now a common scenario. Examples of this in clustering can be found in the bioinformatics field: for instance, in gene expression analysis, bio-signal processing, and of course in other fields such as text clustering.

One of the main issues with the above scenarios is the *curse of dimensionality* (Bellman, 1957), which we discussed in Chapter 3, Section 3.1. This states that a linear increase in the number of features leads to an exponential increase in a hypothesis space (Liu and Modota, 2008). Kriegel et al. (2009) identify other issues resulting from a high-dimensional space such as the effect this space can have in some distance measures used in conventional clustering algorithms, also discussed in Section 3.1; the fact that it is impossible to visualise problems in this type of space and by consequence it is difficult for us to think in it; the irrelevance of some of the features; and that as there are a large number of features, some may be correlated. Irrelevant or correlated features may have a major impact on clustering.

It is difficult to set a hard rule to define how many features are necessary for a dataset to be considered high-dimensional. In fact, the difficulty in clustering such datasets also depends on the data itself, as for certain datasets 30 features may be considered high-dimensional.

Hand et al. (2001:193) describe the curse of dimensionality from a different point of view. They state that the amount of data needed to maintain a specific level of accuracy tends to increase exponentially in relation to the dimensions of a dataset.

From this point of view, a possible worst-case scenario for clustering would be one in which the number of features is much larger than the number of entities and the data itself has clusters that are not easily separable.

In Chapters 3 and 4, we explored the behaviour of WK-Means and iMWK-Means, among others, in datasets with as many as 75 features. In this chapter, we take this experiment a step further and enter the realm of datasets with thousands of features. We have decided to use datasets with a far more complex structure than the Gaussian models that were used previously, choosing instead to cluster Electroencephalography (EEG) signals.

The literature widely acknowledges that EEG signals contain information about the present state or intention of the mind (Drongelen, 2007; Geng et al., 2008; Sanei and Chambers, 2007; Ungureanu et al., 2005; Lee et al., 2005). The acquisition of this type of signal falls into the category of non-invasive approaches and can be used to gather information for mental-task separation.

There are multiple reasons to analyse EEG signals. Some researchers see them mainly as an alternative channel of communication, which could be used for instance by motor-paralysed people who have intact cognitive and sensory functions. Other scholars have different aims, such as diagnosing mental disorders and other abnormalities that may happen in a human body (Ungureanu et al, 2004), or among other possibilities, monitoring the depth of anaesthesia (Ortolani et al., 2002).

The processing of EEG signals is non-trivial, and patterns tend to be difficult to discern. The reason for this is unlikely to be related solely to the high dimensionality of the data, as research suggests that high-dimensional data are of simple structure (Murtagh, 2009). Aside from the fact that we do not fully understand the inner workings of the brain, another point to note is that EEG data is particularly noisy. The

reasons for this noisiness are many, but most of it tends to come from within the brain or on the scalp: for instance, these signals are normally acquired via electrodes put on the scalp of a subject, each of which simultaneously captures signals from thousands of neurons. Other sources of noise include eye movement, muscle activity, cardiac activity, respiration, etc. (Sanei and Chambers, 2007:10). In fact, even if the pure biological EEG source were to be noise-free, the processes of amplification and digitalisation of the signal (also known as systematic bias) would add this (Drongelen, 2007:35). The exposure to pulse modulated electromagnetic fields, which occurs while using devices such as mobile phones, may also generate irregularities in EEG patterns, as research suggests (Klitzing, 1995; Huber et al., 2002), since this may affect cerebral blood flow in certain areas of the brain.

EEG data is high dimensional by nature. A recording of one hour using 128 electrodes at 500 samples per second would generate around 0.45GB of data (Sanei and Chambers, 2007: 14). Although we use only five bipolar electrodes here, the datasets we introduce in Section 5.2 have 5,680 dimensions.

5.2 Setting of the experiment with all features

In this chapter, we experiment with WK-Means and iMWK-Means: both algorithms can be found in Chapter 3, Sections 3.2.1 and 3.3.3 respectively. Our objective is to find these algorithms' maximum accuracy in a high-dimensional space where the number of features is far greater than the number of entities and the data is difficult to cluster.

The data used in this experiment comes from three healthy subjects and was recorded using five bipolar electrodes, generating five channels and a sampling

frequency of 250 Hz. Although the literature tends to use considerably more electrodes, in the order of 32, 64 or even 128, we have decided to follow the example of Tsui et al. (2009) and use a reduced number to minimise computational effort. Our five electrodes were placed according to positions standardised in the extended 10-20 system using fc3 to pc3, fc1 to pc1, cz to pz, fc2 to pc2, and fc4 to pc4.

In each trial, the subject was required to imagine a single task, generating data which was recorded for eight seconds and had 2,001 samples. The tasks could be either:

(T1) Movement of the left hand, or

(T2) Movement of the right hand, or

(T3) Movement of the feet.

The data obtained in a trial was recorded as a three-way function $a=f(t, c, r)$, in which a represents the amplitude, t corresponds to a sample and ranges from one to 2,001, c to one of five channels c=1,…,5, and r to a trial. The three subjects from which the data was collected are denoted as *A*, *B* and *C* and had 240, 120 and 350 trials, respectively; there is no particular reason for the difference in the number of trials. In our experiment, we aimed to cluster the trials (entities) into the right tasks (clusters).

EEG patterns are normally found in frequencies rather than amplitudes (Drongelen, 2007: 3). Taking this issue into account, we have transformed the raw data into its power spectrum density (psd, also known as spectral density or energy spectral density) as it helps to identify periodicities and was successfully applied in EEG signal processing in previous publications including our own (Amorim et al., 2010; Gan, 2006; Chiappa and Bengio, 2004; Millan and Mourino, 2003).

This process represents a trial as a function of time and frequency channel. Each trial generated 71 samples relating to time and 80 psd-features for each of the samples;

the latter consisted of psds over 8~45Hz (psds in 16 frequency bands from each channel: we had five channels hence 80 psd-features). The number of features for clustering represents the 71 time samples and the 80 psd-features; consequently, each trial had 5,680 (71x80) features. It is interesting to note that this number is far greater than the number of trials: 240,120 and 350 for subjects *A*, *B* and *C*, respectively.

All of the datasets were normalised in the same way as described in Chapter 3, Section 3.4. Unlike some of the datasets presented in Chapter 3, however, those present here were entirely comprised of numerical features. The data per subject over their two principal components can be visualised in Figures 5.1, 5.5 and 5.9. Unsurprisingly, the tasks did not appear to separate well.

5.3 Experimental results with all features

In this section, we analyse the results per subject. For each subject, we will firstly deal with the accuracies obtained by each algorithm with its respective optimal β and then analyse the obtained feature weights in each algorithm.

(1) Subject *A*

The normalised psd data for subject *A*, on the plane of its first two components, can be visualised in Figure 5.1. There seems to be a cluster structure in this dataset, but it does not coincide with the given labels; these labels are represented by different shapes.

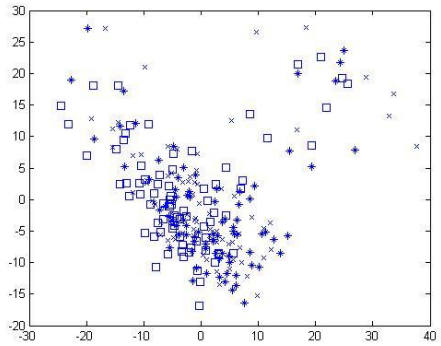

Figure 5.1: PSD data for subject *A* on the plane of the first two principal components, the clusters are shown using different shapes for the trials

The accuracies obtained in each of the algorithms can be found in Table 5.1. Subject *A* had 240 trials divided equally between the three clusters, each of which had 80 trials.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	4.3	41.5	5.9	52.5
iMWK-Means	2.0	2.0	50.1		
	3.0	3.0	35.0		
	4.0	4.0	35.0		
	5.0	5.0	34.6		

Table 5.1: Accuracies achieved using WK-Means and iMWK-Means clustering at the EEG data for subject *A*

The evolution of the accuracy per β for subject *A* can be seen in Figure 5.2. The figure shows that iMWK-Means' accuracy tends to be below 42%, but with a spike to

just over 50% when β =2. The average accuracy of WK-Means per β seems more stable, but is low, mostly at around 38% but with a maximum of 41.5%.

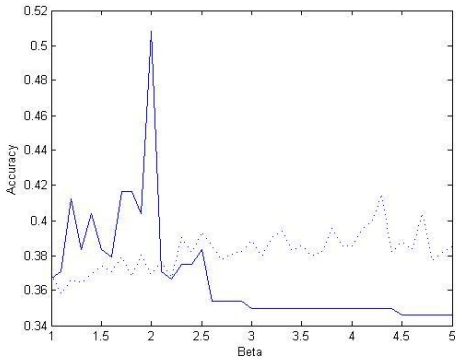

Figure 5.2: Accuracy of iMWK-Means and WK-Means per beta in subject *A*'s data. IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively

In terms of weights, in the anomalous pattern step of iMWK-Means, the three means obtained by averaging the weights over the features are very similar, but their range is not: for instance, the range of weights of the third cluster is twice as much as that of the first cluster. The final weights obtained in iMWK-Means, unlike those obtained with WK-Means, seem to identify some weights as being more information-carrying than others. Figure 5.3 is a bar chart depicting the mean of the weights per feature.

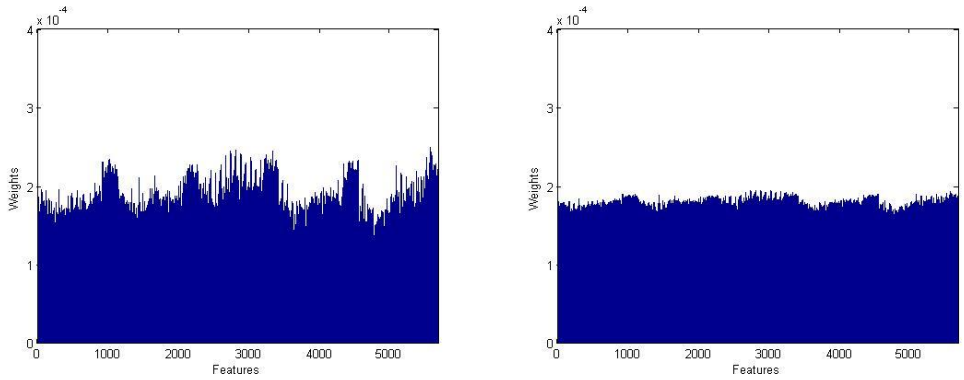

Figure 5.3: Mean of the three final weights (one per cluster) per feature for subject *A*. IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

The pattern of the final weights of iMWK-Means shown in Figure 5.3 seems to be linked to the five channels in the data. These channels are represented in the features, in blocks of 1,136 (the number of time points: 71, times the number of frequency bands from each channel: 16). In terms of time, the weights seem to be higher in the second half of the trials, which conforms to the literature. Figure 5.3 shows that the weights tend to be higher at the end of each of the blocks of 1,136 features.

Another interesting fact we can identify is the relation between the standard deviation of the mean of weights over the clusters and β. It seems that the larger the β, the smaller the standard deviation of weights, for both iMWK-Means and WK-Means, as if the weights were converging. Figure 5.4 shows this relationship in subject *A*'s data.

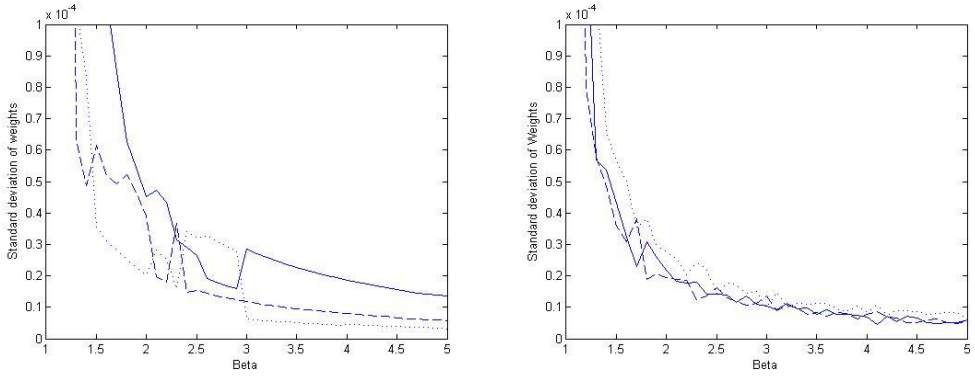

Figure 5.4: Standard deviation of weights per beta on subject *A*: the solid, dashed and dotted lines represent the clusters for tasks T1, T2 and T3, respectively; the figure of iWMK-Means is on the left while WK-Means is on the right

(2) Subject *B*

We begin the analysis of subject *B*'s data by showing the normalised data itself, on the plane of the first two principal components, in Figure 5.5. The different shapes relate to the different tasks.

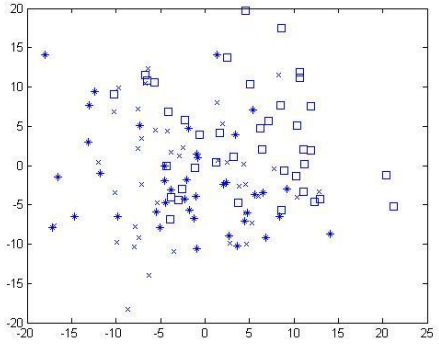

Figure 5.5: PSD data for subject *B* on the plane of the first two principal components, the clusters are shown using different shapes for trials

The accuracies obtained in each of the algorithms are recorded in Table 5.2. Subject *B* has 120 trials, divided equally between the three clusters.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	2.6	45.5	5.9	58.3
iMWK-Means	2.1	2.1	50.1		
	3.0	3.0	47.5		
	4.0	4.0	50.0		
	5.0	5.0	50.0		

Table 5.2: Accuracies achieved using WK-Means' and iMWK-Means' clustering at the EEG data for subject *B*

The accuracy provided by iMWK-Means, per β, was mostly around 48% to 50%. The algorithm demonstrated a tendency to stabilise at 50% at higher βs. The average accuracy of WK-Means was considerably more modest, and evidence of this can be seen in Figure 5.6.

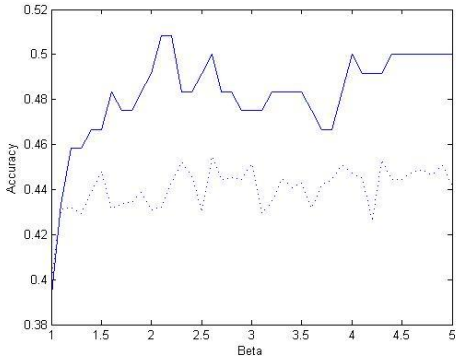

Figure 5.6: Accuracy of iMWK-Means per beta in subject *B* data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively

The first set of weights obtained in the anomalous pattern step of iMWK-Means was equal with a value of 1/5,680 (the total number of features). This signals, again, the difficulty this mechanism may face when dealing with high-dimensional data. These identical weights were then used in the initialisation of iMWK-Means' second step which made it, regarding weights, very similar to MWK-Means.

The final weights obtained by iMWK-Means show a more modest pattern than for subject *A*. WK-Means weights' are closer to the mean, but this time we can detect some level of similarity to those obtained with iMWK-Means. Figure 5.7 shows a bar chart with the means of the weights per feature for subject *B*.

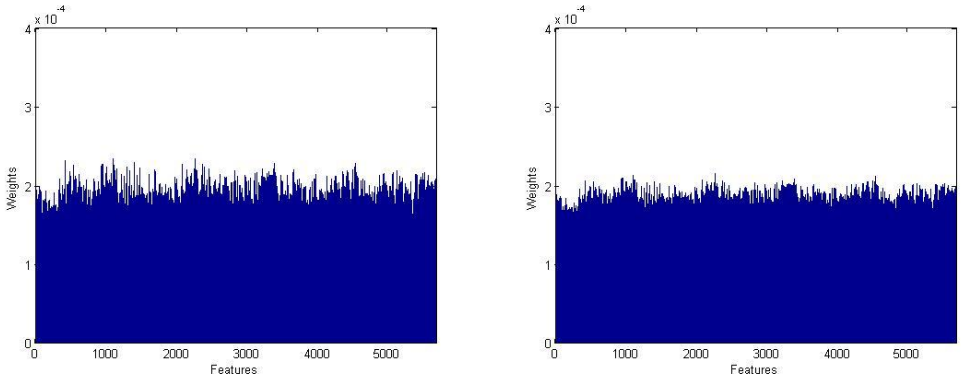

Figure 5.7: Mean of the three final weights (one per cluster) per feature for subject *B*: IMWK-Means and WK-Means are shown in the left and right figures, respectively

We can observe that the standard deviation of the weights over the features seems, again, to be inversely proportional to the value of β, and the higher the β the smaller the standard deviation. The rate of change seems to be rather similar for all three clusters as shown in Figure 5.8.

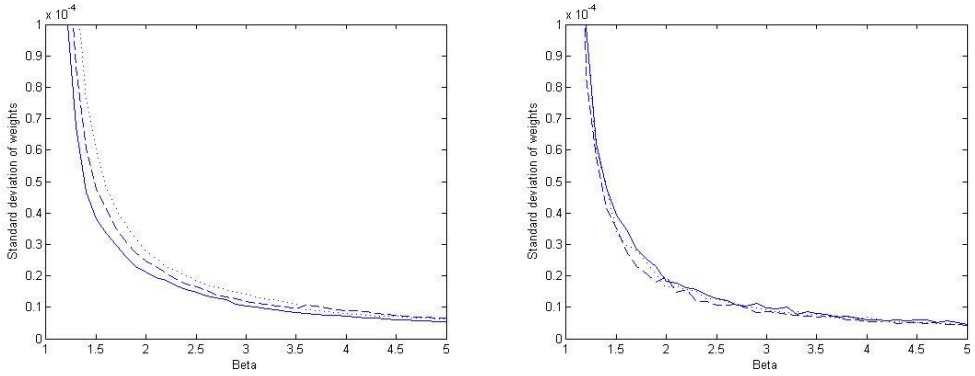

Figure 5.8: Standard deviation of weights per beta on subject *B*. The solid, dashed and dotted line represent the clusters for tasks T1, T2 and T3, respectively: the figure of iWMK-Means is at the left while WK-Means is on the right

(2) Subject *C*

We begin the analysis of subject *C* by showing the normalised data on the plane of the first two principal components in Figure 5.9.

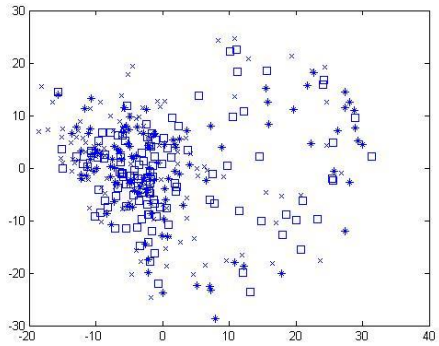

Figure 5.9: PSD data for subject *C* on the plane of the first two principal components, the clusters are shown using different shapes for trials

Table 5.3 shows the accuracies obtained for each of the algorithms. Subject *C* had 350 trials divided into 119, 113, and 118 trials for tasks T1, T2 and T3, respectively.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	1.9	37.3	1.66	45.7
iMWK-Means	1.5	1.5	38.9		
	3.0	3.0	36.3		
	4.0	4.0	35.4		
	5.0	5.0	36.3		

Table 5.3: Accuracies achieved using WK-Means' and iMWK-Means' clustering at the EEG data for subject *C*

Neither iMWK-Means nor WK-Means successfully clustered subject *C*'s data, achieving results that were slightly over random. Figure 5.10 shows the accuracy of both algorithms per beta for this subject.

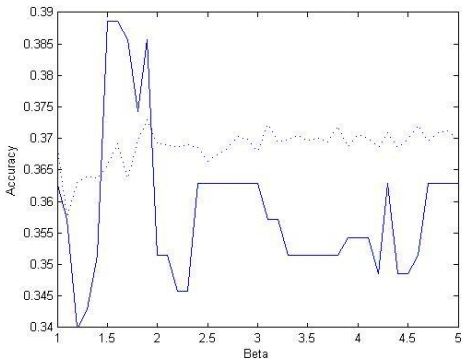

Figure 5.10: Accuracy of iMWK-Means per beta in subject C data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively

As for subject *A* WK-Means had a more stable result than iMWK-Means, perhaps because the former is a multiple-run algorithm, but again it had a lower accuracy.

The set of weights generated at the anomalous pattern step of iMWK-Means, as in the previous subjects, had similar means for the three clusters over the features. Their ranges were again different but did not follow the same pattern as for subject *A;* this time, the ranges were larger in the clusters for tasks T2 and T1 and T3, respectively.

Figure 5.11 shows the time-related pattern of the final mean of weights over the clusters, obtained by iMWK-Means.

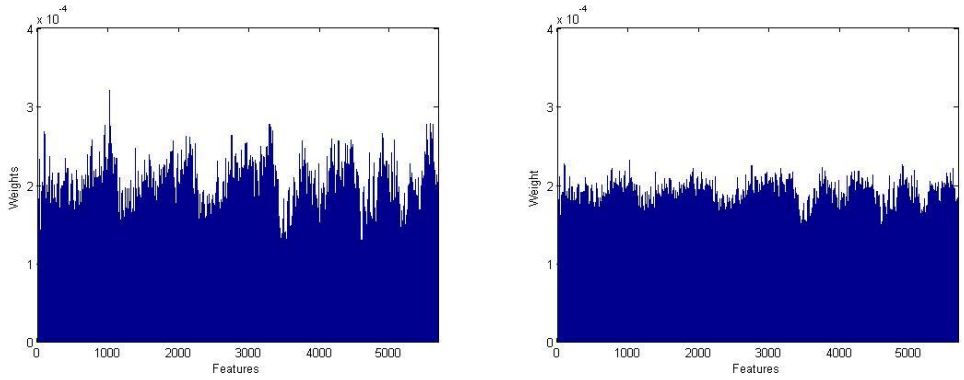

Figure 5.11: Mean of the three final weights (one per cluster) per feature for subject *B*: IMWK-Means and WK-Means are the left and right figures, respectively

Figure 5.12 shows that, as with the previous subjects (Figures 5.4 and 5.8), subject *C*'s data presents a relationship between the standard deviation of weights over the features and the exponent β. Again, this seems to be inversely proportional in the sense that the higher the β, the smaller the standard deviation of weights over the features.

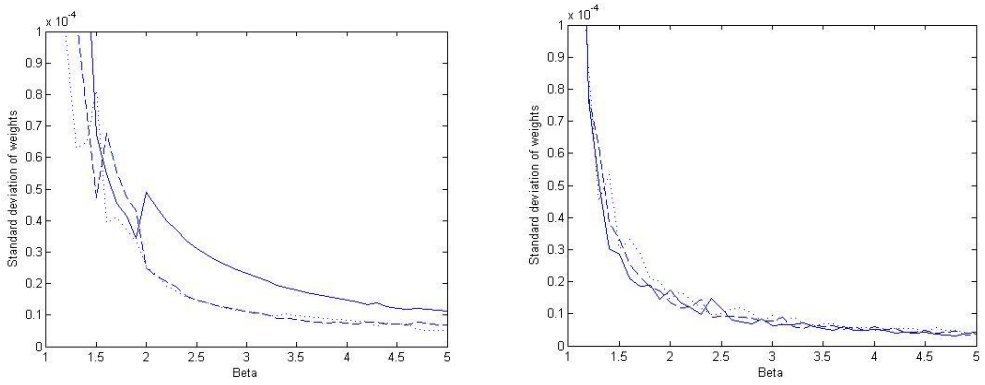

Figure 5.12: Standard deviation of weights per beta on subject *C*: The solid, dashed and dotted line represent the clusters for tasks T1, T2 and T3 respectively; the figure of iWMK-Means is at the left while WK-Means is at the right

In a previous publication (Amorim et al, 2010) we presented a supervised method for classifying mental tasks that achieved 67%, 84% and 82% accuracy for subjects *A*, *B* and *C*, respectively. Of course, we cannot hope to achieve such high accuracies with an unsupervised algorithm but these accuracy rates do suggest the need to analyse the shortcomings of weighted clustering methods.

Weighted methods, such as iMWK-Means and WK-Means, have the following weaknesses:

(i) If a set of features carries similar meaningful information, none will be excluded (assigned a weight zero); in fact, they will all end up with similar weights.

(ii) Even if a feature weight has a negligible value, computers will have to take this into account when making their calculations.

(iii) They analyse one feature at a time so are unable to deal with the XOR problem.

In relation to problem (i), Guyin and Elisseeff (2003) suggest that feature selection (and we extend this here to feature weighting) tends to be suboptimal for building a predictor in datasets with a high level of redundancy. If a set of features

carries similar information, there is no need for the algorithms we deal with in this thesis to take them all into account.

Problem (ii) relates to computational effort. It is intuitive that features should not have negligible weights. Unfortunately, problem (iii) is difficult to solve because the weights' algorithms presented in Chapter 3 follows a wrapper approach and calculate weights by taking into account one feature at a time. Therefore, we will leave this for future research.

Issues (i) and (ii) lead us to believe that algorithms such as WK-Means and iMWK-Means would benefit from a feature-space reduction in the scenario presented in this chapter.

5.4 Feature-space reduction

Huang el al. (2008) suggests that WK-Means can be used for feature selection. In their explanation, they follow a simple method with three steps:

1- Running WK-Means on the dataset or a subsample of it;

2 - Removing those features that have a small weight;

3 - Running WK-Means, or a different clustering algorithm, on the dataset with the remaining features only.

At first, the above method could seem to reduce computations, and even to deal with the issue of features with negligible weights. Unfortunately, Huang el al. (2008) does not clearly define the smallness of the weight. If we were to follow their method, we would need to introduce two extra thresholds as parameters: one to define what

constitutes a small weight and other to be used as a stop condition. We would rather not introduce two extra parameters as this could increase the computational effort involved in finding them.

Huang et al. (2008) acknowledge that in large datasets several runs of WK-Means may be necessary, and this may lead to an even greater computational effort. WK-Means is a multiple-run algorithm; in all our experiments, we have used 100 runs as a standard. In order to follow their method we would have to run it 100 times per β, remove the features with small weights (i.g the bottom 10%) and run it another 100 times per β, and so forth, thus creating a computationally-demanding loop.

In Section 5.4.1, we present Mitra et al.'s (2002) unsupervised feature-selection method that deals with redundant features and needs only one extra parameter. We extend this in Section 5.4.2 by introducing a new method. In our method the number of extra parameters is reduced to zero.

5.4.1 Feature selection using feature similarity

Mitra et al. (2002) have introduced what is arguably the most popular unsupervised method for dealing with redundant features. Their method, feature selection using feature similarity (FSFS), does not present a search process and by consequence is said to be computationally undemanding. The core of their method involves removing redundant features by determining a set of maximally independent features and then discarding similar ones. They begin by presenting the correlation coefficient that is then used in their introduced index.

$$p(x,y) = \frac{cov(x,y)}{\sqrt{var(x)var(y)}}$$

Equation 5.1: Correlation coefficient

in the above equation, *var* and *cov* represent the variance and covariance between two variables, respectively. Mitra et al. (2002) then present the maximal information compression index (λ_2), which is the smallest eigenvalue of the covariance matrix of random variables *x* and *y*. It measures *the minimum amount of information loss* or *the maximum amount of information compression* possible. They acknowledge that the definition of this measure is not entirely new, but its use in unsupervised feature selection is innovative.

$$2\lambda_2(x,y) = (var(x) + var(y) - \sqrt{\left(var(x) + var(y)\right)^2 - 4var(x)var(y)(1 - p(x,y)^2)}$$

Equation 5.2: Maximal information compression index

λ_2's value is zero when the features are linearly dependent, which increases as the independence decreases. This index has the following properties (Mitra et al., 2002):

(i) $0 \leq \lambda_2(x,y) \leq 0.5(var(x) + var(y))$

(ii) $\lambda_2(x,y) = 0$, iff *x* and *y* are linearly related

(iii) $\lambda_2(x,y) = \lambda_2(y,x)$ (symmetric)

(iv) It is invariant to translation of the dataset

(v) It is invariant to rotation.

Although not completely bound to it, the FSFS algorithm (Mitra et al., 2002) uses the maximum information compression index to select features: this can be found in Algorithm 5.1.

1. *Initial setting*

   Choose an initial value for *k*, which has to be smaller or equal to the number of features, minus 1. Put all features in the reduced subset *R*. Standardise the features rather than the entities.

2. *Calculate the dissimilarity*

   For each feature $F_i \in R$ calculate r_i^k, the dissimilarity between F_i and its *k*th nearest neighbour feature in *R*, using the maximal information compression index (Equation 5.2).

3. *Find the feature to be retained*

   Find the feature $F_{i'}$ for which $r_{i'}^k$ is the minimum. Retain $F_{i'}$ and discard its *k* nearest features. Set $\varepsilon = r_{i'}^k$

4. *Adjust k in relation to the number of features*

   If *k* > cardinality(*R*) -1, then *k* = cardinality(*R*) -1

5. *Stop condition*

   If *k* =1, stop and output the reduced feature set.

6. *Adjust k in relation to the similarity*

   While $r_i^k > \varepsilon$, do

   6.1 *k* = *k* -1.

   $r_i^k = inf_{F_i \in R} r_i^k$

   6.2 if *k*=1 go to step 5

   Go to step 2.

Algorithm 5.1: Unsupervised feature selection using feature similarity

In Algorithm 5.1, step 2, the use of the maximal information compression index (Equation 5.2) is not compulsory; instead the dissimilarity may be calculated using other indexes, such as the correlation coefficient (Equation 5.1): here we use the former.

Mitra et al. (2002) have conducted a number of experiments comparing their FSFS algorithm with others. Their method was successfully compared to a number of well-known algorithms, using different comparison indices: Brand and Bound's algorithm (Devijver and Kittler, 1982); sequential forward search (Whitney, 1971; Devijver and Kittler, 1982); sequential floating forward search (Pudil et al., 1994); stepwise clustering, using the correlation coefficient (King, 1967); and the supervised Relif-F (Kononenko, I. 1994).

5.4.2 Feature selection through iKFS

The FSFS algorithm does not take into account the cluster structure of the features and requires an extra parameter, these issues leads us to analyse a possible K-Means based extension. The use of K-Means for clustering features has been suggested in a number of publications (Dhillon et al., 2003; Guyon and Elisseeff, 2003; Dy, 2008). It is an appealing method as if features are clustered together they can be considered similar, while those in different clusters are dissimilar. The main idea is to replace a whole cluster of similar features by one feature only: normally, the cluster centroid (Guyon and Elisseeff, 2003).

In the clustering of features, we have decided not to use a weighted method such as WK-Means or iMWK-Means, for the following reasons:

(i) Weighted methods excel when different features have different degrees of relevance for clustering. We will cluster the features themselves; hence the weights will be given to the trials, which we assume to be of equal relevance.

(ii) This would increase the number of parameters needed. Both algorithms, WK-Means and iMWK-Means, rely on a good β. One β would be used to cluster the features, and probably a different one would then be used when clustering the trials.

The second possibility would be to use either K-Means or iK-Means (presented in Section 2.4, Algorithm 2.6). Although the latter did not appear to cope well when integrated with WK-Means and the Minkowski metric in a high-dimensional space, the experiments in Section 3.5 show it as a rather competitive to K-Means itself.

Potentially the greatest advantage of using iK-Means as a starting point is that it finds clusters in data. This allows us to create a new method for feature selection via clustering features that has no unspecified parameter. If we were to use K-Means, we would have to set the number of clusters in the data as a pre-specified parameter. The reduced computational effort generated by iK-Means is also of interest as a deterministic method.

In our approach, which we will call iK-Means with feature similarity (iKFS), we cluster the features using iK-Means and set its *discarding threshold* of zero (for clarification, see Algorithm 2.6, in Chapter 2). This configuration allows iK-Means to find clusters composed of a single feature. It does not seem intuitive to represent clusters of different sizes by one feature each, so the number of features selected from each cluster depends on its size and the total number of clusters in the dataset, as per Equation 5.3.

$$f_k = \left\lceil \frac{Cardinality(S_k)}{Cardinality(S)} K \right\rceil,$$

Equation 5.3: Defining the quantity of features *f* selected from a cluster

where the number of features f_k selected from a cluster S_k is equal to the cardinality of cluster S_k(the number of features within this cluster) found in the anomalous pattern step, divided by the total number of features, times the total number of clusters *K*, rounded up. Algorithm 5.2 formally introduces our method.

1. *Initial setting*
   Standardise the features, rather than the entities, using Equation 3.13.
2. *Cluster*
   Find the clusters of the features using iK-Means, Algorithm 2.6. The iK-Means threshold should be set to zero, allowing it to find singletons.
3. *Feature selection*
   For each cluster, find the *f* features as per Equation 5.3 that have the largest maximal information compression (Equation 5.2) and save them in the reduced subset *R*.
4. *Output*
   Output the subset *R* of selected features.

Algorithm 5.2: iK-Means with feature similarity (iKFS)

Our method also allows the use of measures other than the maximal information compression (Equation 5.2). In fact, in our early experiments we tried to use the correlation index (Equation 5.1) but were not as successful or fast. The speed was particularly low when trying to find a new centroid, an entity with the highest average absolute correlation to all other entities in the same cluster. In our early experiments we also sought to set *f* to a constant, such as one or five, but the results for this were not as good as those revealed in Section 5.5.2.

Through experimentation, we found iKFS (Algorithm 5.2) to be considerably faster than FSFS (Algorithm 5.1).

5.5 Experimental results with the reduced datasets

In Sections 5.5.1 and 5.5.2, we present the results of our experiments with the FSFS method introduced by Mitra et al. (2002) and our iKFS method.

As we shown in algorithm 5.1, the FSFS method calculates a dissimilarity index between all features and their respective *k*th nearest neighbour, by consequence it increases the number of parameters we need in one, the *k*. Our objective in the following experiments is not to find this *k* but rather to compare the optimal accuracies obtained with WK-Means and iMWK-Means with good *k* and β and compare them to the accuracies obtained with our iKFS method, which does not require a new parameter.

We have run a number of experiments to find a good *k*; these were in the range of 4,800 to 5,600, in steps of 100. For each of the generated subset of features, we applied both WK-Means and iMWK-Means, as before, with β ranging from 1.0 to 5.0 in steps of 0.1. In our experiments with iMWK-Means, we consider *k* to be optimal

when the maximum accuracy using all possible βs with this k is higher than the maximum accuracy obtained with all other k. Experiments with WK-Means were very similar, but as this is a multiple-run algorithm we used the maximum of the average accuracies per β.

5.5.1 Experiments with the FSFS method

In this section, we show the optimal accuracy obtained with both WK-Means and iMWK-Means using only the features selected by the FSFS method (Mitra et al., 2002). Note that the amount of features is not necessarily the same for WK-Means and iMWK-Means, as these may have different optimal ks.

(1) Subject *A*

We have applied principal component analysis to subject *A*'s data, using only the 28 features selected for iMWK-Means and the 25 selected for WK-Means. Figure 5.13 shows these two datasets on the plane of their first two principal components as the top-left and top-right figures, respectively. For comparison, the bottom figure shows the dataset on its first two principal components using all features.

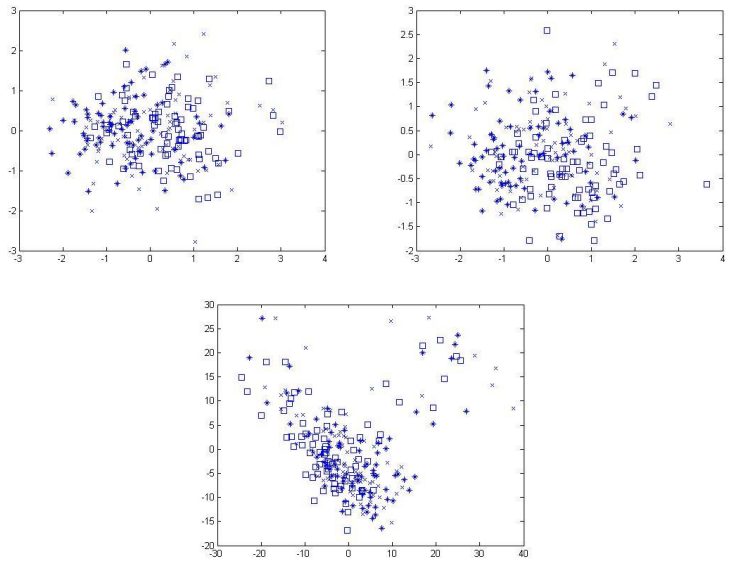

Figure 5.13: PSD of subject *A* on the plane of the first two principal components, the clusters shown using different shapes for the trials: the top-left and top-right figures use 28 and 25 features, respectively, selected with the FSFS method; the figure at the bottom uses all features

It is difficult to see much improvement, in particular for the features selected for iMWK-Means, in relation to having all of the features present in the dataset. This is further validated by the accuracies in Table 5.4.

Algorithm	Optimal *k*	Exponent β at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
WK-Means	5000 (25)	2.0	3.9	48.2	2.5	52.1
iMWK-Means	5300 (28)	3.2	3.2	51.2		
		3.0	3.0	47.5		
		4.0	4.0	47.5		
		5.0	5.0	46.7		

Table 5.4: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the features selected using FSFS, on subject *A*'s data; the number of features is shown in parentheses

Unfortunately, the optimal value for iMWK-Means does not seem to have had a statistically significant improvement than if no feature selection had been performed. WK-Means, on the other hand, did present an increase of nearly 7% in its mean value but had no positive difference in its maximum. We further compare both algorithms in Figure 5.14, showing its accuracies per β using their respective optimal *k*s.

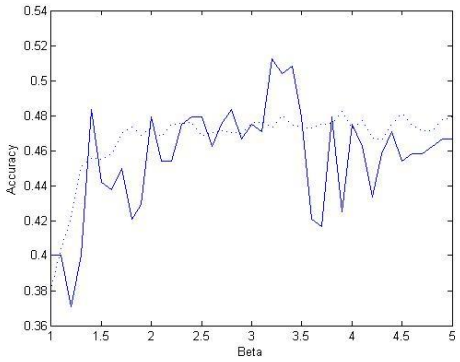

Figure 5.14: Accuracy of iMWK-Means per Beta in subject *A*'s data. IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively: the above was obtained using only the features selected via the FSFS method for each algorithm

In terms of the final weights, both algorithms were very similar in all features, as shown in Figure 5.15.

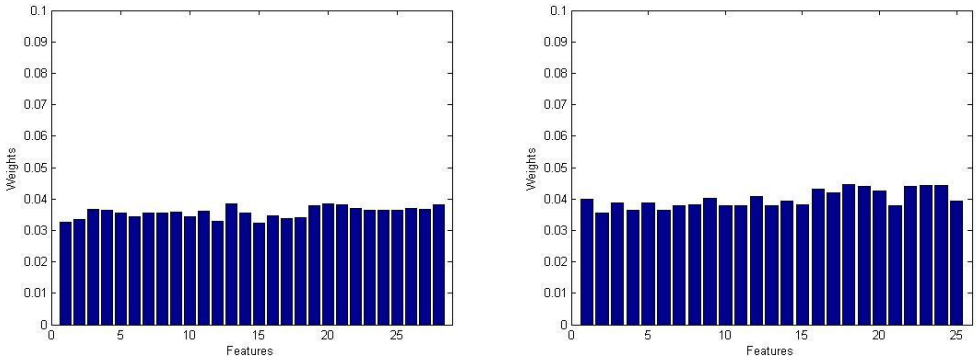

Figure 5.15: Mean of the three final weights (one per cluster) in each of the features selected via FSFS for subject *A*: IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

(2) Subject *B*

As for the previous subject, we have applied principal component analysis to this subject's data using only the features selected with FSFS, 393 and 472 for iMWK-Means and WK-Means, respectively. Figure 5.16 shows these two datasets on the plane of their first two principal components in the top-left and top-right. Again, we have also at the bottom the original dataset on its first two principal components using all features.

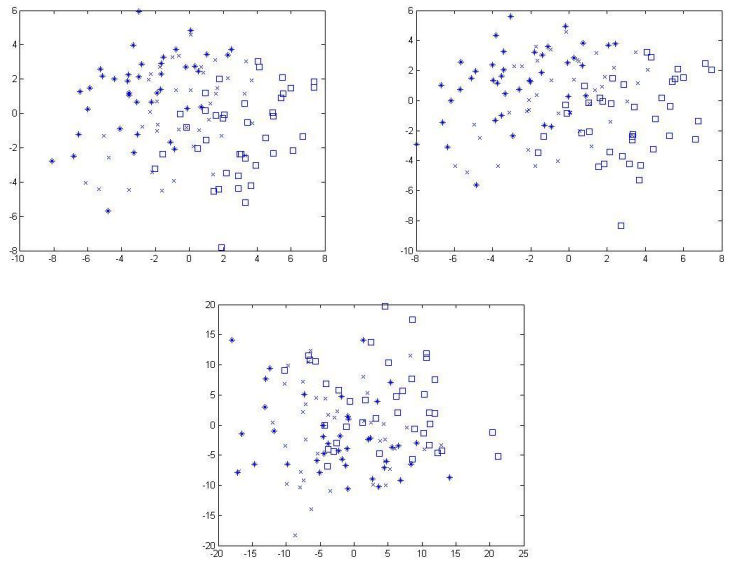

Figure 5.16: PSD of subject *B* on the plane of the first two principal components, the clusters are shown using different shapes for trials: the figures in the top-left and top-right use 393 and 472 features, respectively, selected with the FSFS method: the figure at the bottom uses all features

On this occasion, there seemed to be some improvement represented in the considerable increase of accuracy for both subjects, as shown in Table 5.5.

Algorithm	Optimal *k*	Exponent β at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
WK-Means	4800 (472)	2.0	4.7	59.2	4.6	73.3
iMWK-Means	5000 (393)	2.0	2.0	65.0		
		3.0	3.0	60.0		
		4.0	4.0	63.3		
		5.0	5.0	52.5		

Table 5.5: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the features selected using FSFS, on the data of subject *A*: the number of features is shown in parentheses

The maximum accuracy of WK-Means is reasonably higher than the accuracy obtained with iMWK-Means, but again its average accuracy using its optima β and k is lower. Figure 5.17 shows the accuracies of iMWK-Means as well as the average accuracy of WK-Means per β.

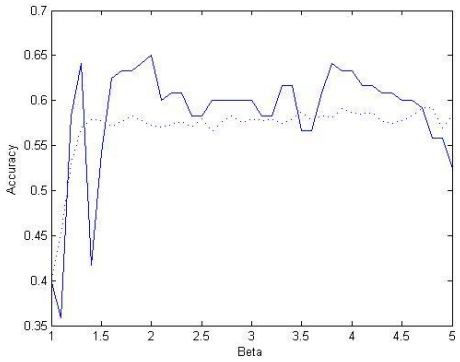

Figure 5.17: Accuracy of iMWK-Means per beta in subject *A*'s data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively; the above was obtained using only the features selected via the FSFS method for each algorithm

It seems that WK-Means has a more stable accuracy. This may have happened because the accuracies are, in fact, averages. Even with a considerably large number of features, the final weights, shown in Figure 5.18, had rather similar values in WK-Means, which may also have added to its stability per β.

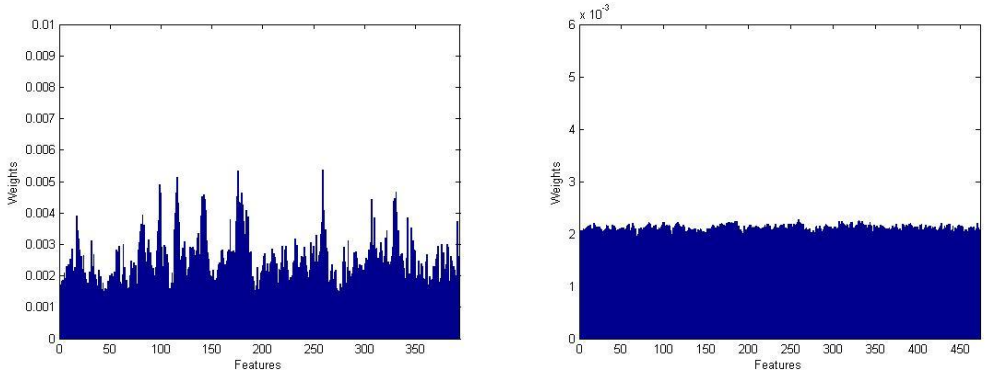

Figure 5.18: Mean of the three final weights (one per cluster) in each of the features selected via FSFS for subject *B*: IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

(3) Subject *C*

This subject had the same optimal *k* for both WK-Means and iMWK-Means; hence in Figure 5.19, we show only the two first principal components of the dataset with the selected 33 features on the left and all of the features on the right.

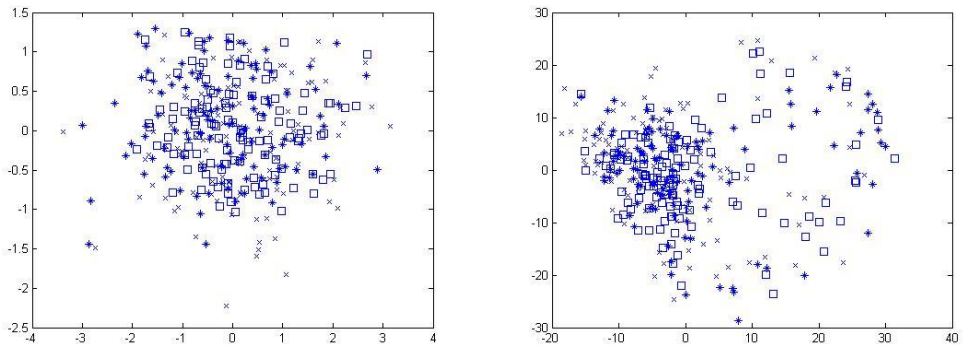

Figure 5.19: PSD of subject *C* on the plane of the first two principal components, the clusters are shown using different shapes for trials: the figure on the left uses only the 33 features selected with the FSFS method, while the one on the right uses all of them

It seems that the data became less sparse, but yet was without a clear cluster structure. Table 5.6 shows that there is, unfortunately, little improvement.

Algorithm	Optimal	Exponent β at		Accuracy, %		
	k	Distance	Weight	Mean	Std	Max
WK-Means	4800 (33)	2.0	4.7	39.6	1.8	42.3
iMWK-Means	4800 (33)	4.6	4.6	42.3		
		3.0	3.0	41.1		
		4.0	4.0	40.0		
		5.0	5.0	38.6		

Table 5.6: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the features selected using FSFS, on subject *A*'s data: the number of features is shown in parentheses

Figure 5.20 shows the accuracies per beta for both algorithms. Again, WK-Means seems to be more stable regarding the β.

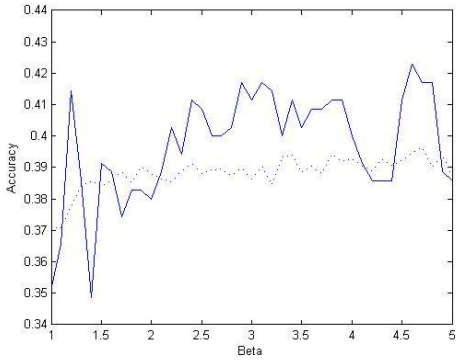

Figure 5.20: Accuracy of iMWK-Means per beta in subject *A*'s data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively; the above was obtained using only the features selected via the FSFS method for each algorithm

As for Subject *A*, the final weights obtained by both algorithms had similar values. This is shown in Figure 5.21.

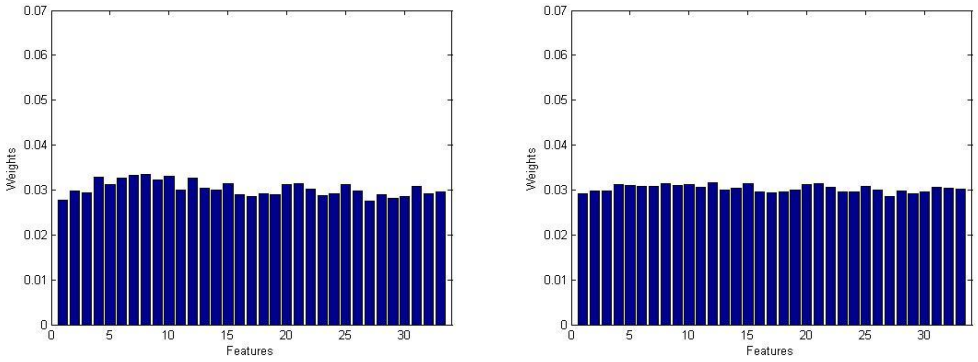

Figure 5.21: Mean of the three final weights (one per cluster) in each of the features selected via FSFS for subject *C*. IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

5.5.2 Experiments with the iKFS Method

In this section, we show the optimal accuracies obtained with both WK-Means and iMWK-Means using only the features selected with iKFS.

(1) Subject *A*

For subject *A*, our method selected only 20 out of 5,680 features. We used PCA in the reduced dataset; Figure 5.22 shows it on the plane of its first two principal components, together with the figure that takes into account all of the features for easy comparison.

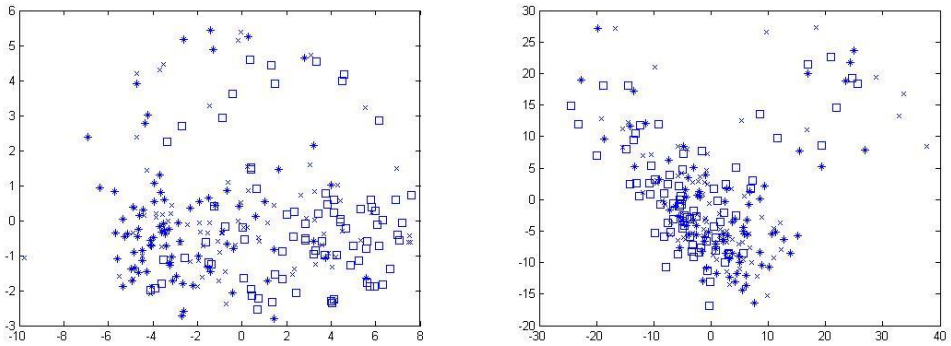

Figure 5.22: PSD of subject *A* on the plane of the first two principal components, the clusters are shown using different shapes for trials: the figure on the left uses only the 20 features selected with iKFS, while the one on the right uses all

Although it is difficult to state from a bi-dimensional figure, it seems that a better cluster structure is now aligned to the x axis. We have applied both WK-Means and iMWK-Means in the reduced dataset and show the accuracies at an optimal β in Table 5.7.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	1.5	54.1	1.4	56.2
iMWK-Means	4.5	4.5	59.2		
	3.0	3.0	56.2		
	4.0	4.0	58.3		
	5.0	5.0	58.7		

Table 5.7: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the 20 features selected via iKFS on subject *A*'s data

In this case, iMWK-Means was consistently more accurate than WK-Means. When using the optimal β, the difference between iMWK-Means and the average accuracy of WK-Means was just over five percent. Figure 5.23 shows a comparison between the accuracies of both algorithms per β.

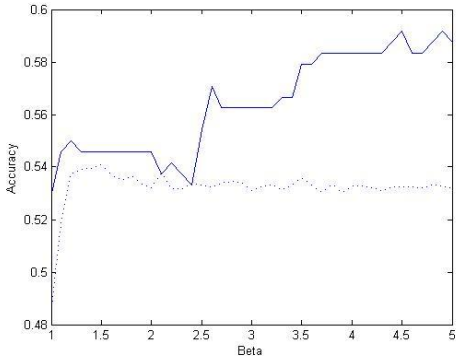

Figure 5.23: Accuracy of iMWK-Means per beta in subject *A's* data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively; the above was obtained using only the 20 features selected via iKFS

The weights found by iMWK-Means were similar to each other on this occasion. WK-means seems to find a considerably bigger difference in weights but, as shown in Figure 5.23, this time this does not seem to yield a greater level of accuracy than that obtained with iMWK-Means.

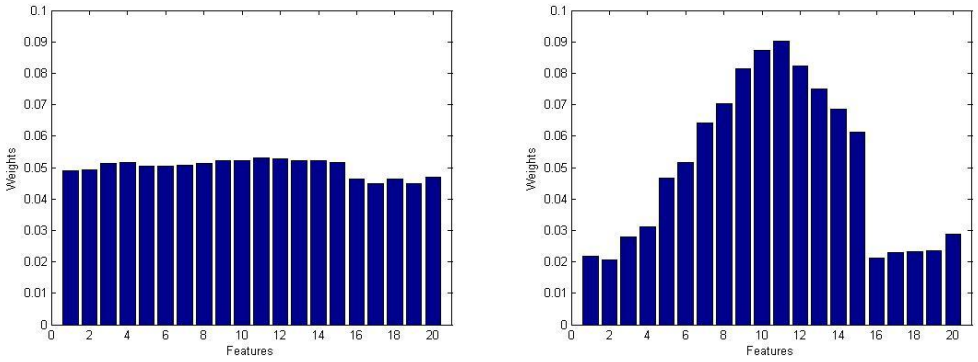

Figure 5.24: Mean of the three final weights (one per cluster) in each of the 20 features selected with iKFS for subject *A*: IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

(2) Subject *B*

Our feature-selection method, iKFS, reduced the dataset from 5,680 features to nine. Figure 5.25 shows the difference between the first two principal components of the reduced dataset (left) and the whole (right).

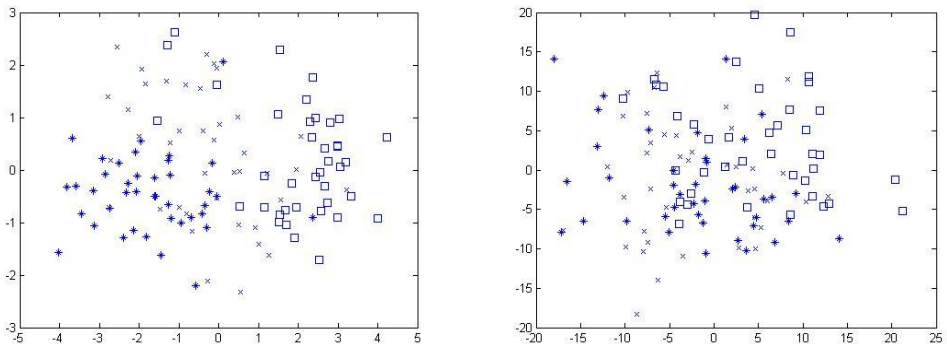

Figure 5.25: PSD of subject *B* on the plane of the first two principal components, the clusters shown using different shapes for trials: the figure in the left uses only the nine features selected with iKFS while the one at the right uses all of them

Again, it seems that the reduced dataset improved the class separability of the trials. This is verified in Table 5.8.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	3.6	68.5	4.7	76.7
iMWK-Means	2.5	2.5	66.7%		
	3.0	3.0	65.0%		
	4.0	4.0	65.0%		
	5.0	5.0	63.3%		

Table 5.8: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the nine features selected via iKFS on the data of subject *B*

In this case, WK-Means had a considerably better maximum accuracy in relation to iMWK-Means. The iMWK-Means algorithm was below but competitive to the average accuracy of WK-Means using their respective optimal βs. Figure 5.26 shows a comparison of the accuracies of iMWK-Means and the average accuracy of WK-Means per β.

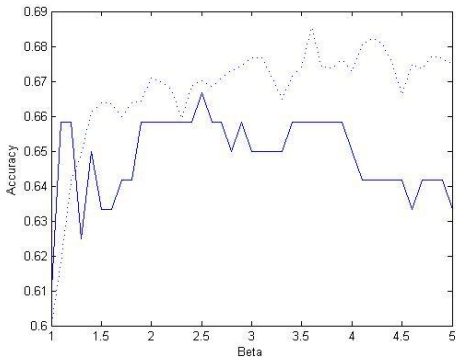

Figure 5.26: Accuracy of iMWK-Means per Beta for subject *B*'s data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively; the above was obtained using only the nine features selected via iKFS

Finally, Figure 5.27 shows the feature weights generated by both algorithms to be similar.

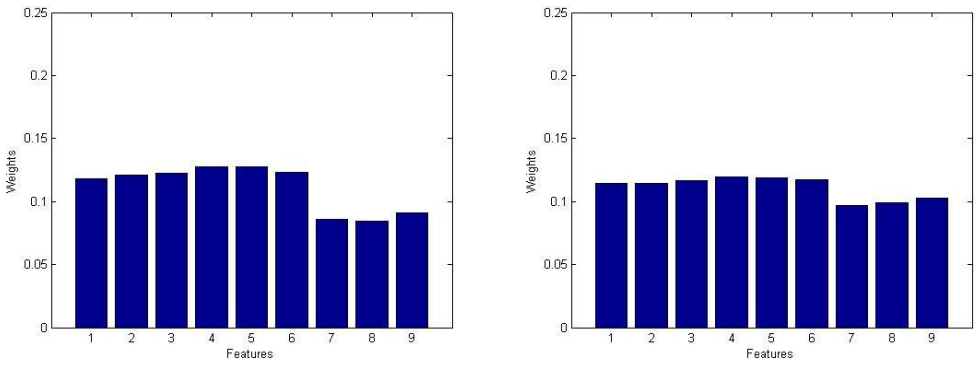

Figure 5.27: Mean of the three final weights (one per cluster) in each of the nine features selected with iKFS for subject *B*: IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

(3) Subject *C*

The iKFS method selected 11 out of a total 5,680 features. Figure 5.28 shows the differences between the first two principal components of the dataset using only the selected features (left) and all of the features (right).

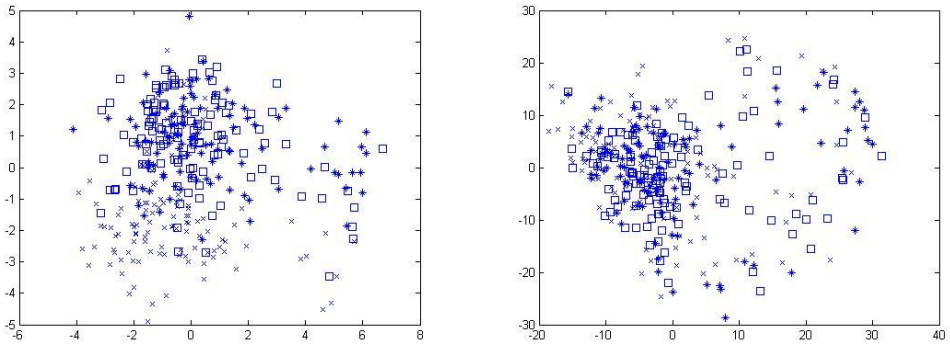

Figure 5.28: PSD for subject *C* on the plane of the first two principal components, the clusters shown using different shapes for trials: the figure on the left uses only the 11 features selected with iKFS while the one on the right uses all of them

Again, the dataset using only the selected features seems to have a better cluster structure than the original with all of them. This is validated with the optimal accuracies shown in Table 5.9.

Algorithm	Exponent β at		Accuracy, %		
	Distance	Weight	Mean	Std	Max
WK-Means	2.0	4.5	56.5	0.3	56.9
iMWK-Means	1.8	1.8	58.6%		
	3.0	3.0	56.9%		
	4.0	4.0	57.1%		
	5.0	5.0	57.1%		

Table 5.9: Accuracies achieved with the WK-Means and iMWK-Means algorithms, using only the 11 features selected via iKFS clustering on subject *C*'s data

IMWK-Means did obtain a higher accuracy than the average and maximal accuracies obtained by WK-Means, but the latter is competitive. To its advantage, iMWK-Means shows a small, but consistent, superiority over WK-Means as shown in Figure 5.29.

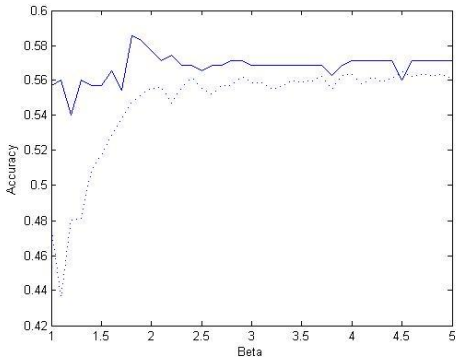

Figure 5.29: Accuracy of iMWK-Means per beta in subject *C* data: IMWK-Means and WK-Means are represented by the solid and dotted lines, respectively; the above was obtained using only the 11 features selected via iKFS

Finally, in terms of feature weights, the ones obtained by WK-Means were similar, while those obtained by iMWK-Means demonstrated a considerable level of difference among certain features, as shown in Figure 5.30.

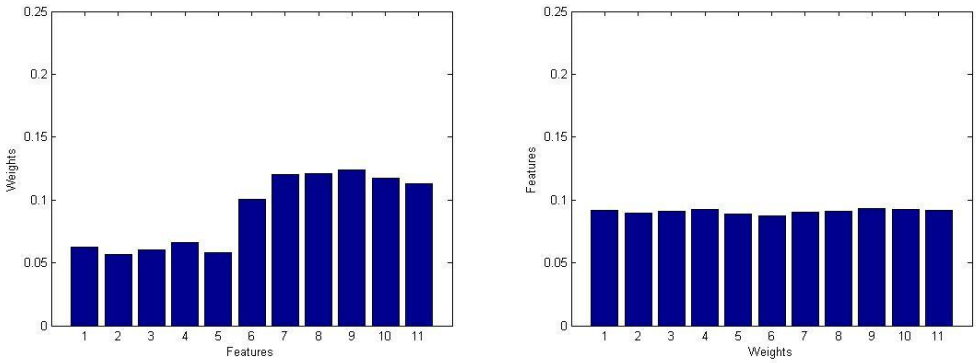

Figure 5.30: Mean of the three final weights (one per cluster) in each of the 11 features selected with iKFS for subject *C*: IMWK-Means and WK-Means are shown in the left- and right-hand figures, respectively

5.6 Comparison of the methods

As expected, both feature selection algorithms FSFS (Mitra et al., 2002) and our feature clustering iKFS generally increased the accuracy of iMWK-Means and WK-Means.

In this section, we compare the results obtained in Sections 5.5.1 and 5.5.2. Our objective is to clarify the differences in the results as much as possible; we start by comparing those obtained for subject *A*, shown in Table 5.10.

Algorithm	Feature Selection Method	Exponent β at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
WK-Means	FSFS (25)	2.0	3.9	48.2	2.5	52.1
WK-Means	iKFS (20)	2.0	1.5	54.1	1.4	56.2
iMWK-Means	FSFS (28)	3.2	3.2	51.2		
iMWK-Means	iKFS (20)	4.5	4.5	59.2		

Table 5.10: Comparison of algorithms for subject *A*; the number of features is shown in parentheses

When comparing the feature selection algorithms, we can observe that the iKFS has enabled WK-Means and iMWK-Means to achieve considerably higher accuracies using a smaller number of features. The iMWK-Means, when using any of the two feature-selection methods, has provided results that are higher than the average accuracy obtained with WK-Means, and competitive or higher accuracy than the maximum achieved by WK-Means.

Table 5.11 shows a similar comparison for subject *B*.

Algorithm	Feature Selection Method	Exponent β at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
WK-Means	FSFS (472)	2.0	4.7	59.2	4.6	73.3
WK-Means	iKFS (9)	2.0	3.6	68.5	4.7	76.7
iMWK-Means	FSFS (393)	2.0	2.0	65.0		
iMWK-Means	iKFS (9)	2.5	2.5	66.7		

Table 5.11 Comparison of algorithms for subject *B*; the number of features is shown in parentheses

Again, in terms of feature selection iKFS has enabled both WK-Means and iMWK-Means to achieve better results using far fewer features than the feature similarity method. IWMK-Means provides a higher or competitive accuracy when compared to the average accuracy of WK-Means using both feature-selection methods, but smaller than WK-Means's maximum. Table 5.12 shows the comparison for subject *C*.

Algorithm	Feature Selection Method	Exponent β at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
WK-Means	FSFS (33)	2.0	4.7	39.6	1.8	42.3
WK-Means	iKFS (11)	2.0	4.5	56.5	0.3	56.9
iMWK-Means	FSFS (33)	4.6	4.6	42.3		
iMWK-Means	iKFS (11)	1.8	1.8	58.6		

Table 5.12: Comparison of algorithms for subject *C*; the number of features is shown in parentheses

The iKFS method seems to provide a much better subset of features than the FSFS method introduced by Mitra et al. (2002). In this case the accuracy obtained with WK-Means is less but competitive to that obtained with iMWK-Means.

5.7 Conclusion

Our objective in this chapter has been to examine the behaviour of both WK-Means and iMWK-Means when dealing with datasets that are set in a high-dimensional space and have classes that are difficult to separate. For this purpose, we decided to cluster EEG data, which is widely acknowledged as a difficult problem. We acquired this data from three healthy subjects, with 5,680 features.

We have analysed the frequency of the datasets by using the power spectrum density of the raw data in our experiments. We have found that iMWK-Means tends to have a better accuracy than the average achieved by WK-Means and the former is, in most cases, competitive to its maximum accuracy with of course less computational effort.

We have also found that reducing large amounts of features that have some level of similarity (are clustered together) to a considerably reduced subset, can have positive outcomes in our scenario. Most likely, this is particularly true for both weighted algorithms because they would equally rate two features carrying the same information, instead of discarding one of them. We have experimented with the FSFS method (Mitra et al., 2002) and developed our own method, iKFS, and found that in our datasets the latter tends to provide better features more quickly and without requiring a new parameter.

In the next chapter we will conclude this work, summarise our contribution to the literature, and clearly define the direction of our future research.

Chapter 6: Conclusion and Future Work

6.1 Research outcomes

The subject of this thesis is to try to further advance into the problem of data clustering, and especially high-dimensional data clustering, through the machinery of feature weighting and selection. We try to take advantage of the popular K-Means clustering method while attempting to reduce the effects of its drawbacks.

First, we extend a recent approach by Huang et al. (2005, 2008) to feature weighting by applying one more, weighting step to the two steps of a K-Means iteration. The extension of the method to Minkowski metric has allowed us to overcome the drawback of Huang at al. approach: the lack of meaning in the parameter β of the method. Moreover, our MWK-Means method has proved in our numerous experiments to be superior to that of Huang et al. However MWK-Means retains the drawback of the generic K-Means, that it needs multiple runs to find a proper solution. To overcome this, we extend Mirkin's anomalous pattern method (Mirkin, 2005:93) to Minkowski metric. We demonstrate that thus modified Minkowski metric K-Means clustering method, iMWK-Means gives a significant reduction of the computation time, while maintaining the competitive advantages of the MWK-Means.

However, to apply this approach to real high-dimensional data, like EEG data one would need one more step in reducing the data dimensionality: the space dimension reduction, ie. Feature selection. To do so we apply the most popular feature selection approach FSFS by Mitra et al. (2002) involving an interesting measure of dissimilarity. However, in our opinion FSFS suffers from drawbacks that should undermine the potential of using the measure. FSFS, first involves a manually specified parameter k and, second it does not take into account the structure of similarities between features.

Therefore, we developed our own version of the approach, algorithm iKFS.

In our experiments, the iKFS method appears to be superior over FSFS indeed. We conclude by applying our developed techniques to the problem of distinguishing between the mental tasks which EEG trial data relate to (our special thanks goes to prof. John Q. Gan for multiple consultations and the data provided with regard to that). Therefore the main results of this project are:

a. Two Minkowski metric based methods, namely Minkowski Weighted K-Means (MWK-Means) and intelligent Minkowski Weighted K-Means (iMWK-Means), which do improve WK-Means's accuracy at original and noisy datasets with different amounts of features and noise features.

b. A method to reduce the feature space of a dataset, intelligent K-Means with feature similarity (iKFS).

c. A machinery for clustering EEG data involving (a) and (b).

6.2 Observations

In our experiments we have being able to observe the following:

d. The anomalous pattern initialisation tends to provide good initial centroids and weights in datasets from modest to moderate number of features.

e. iMWK-Means does not seem to work as accurately as MWK-Means in high-dimensional spaces with Gaussian mixture datasets containing a high amount of noisy features.

f. Minkowski metric-based algorithms tend to have a considerably more stable β.than the original approach by Huang et al.

g. The experiment using Gaussian mixtures with iMWK-Means shows a clear similarity between the estimated optimal β, found using 20% of the data, and the optimal β for the whole dataset.

h. Yet in the same experiment, there is an inversely proportional relationship between the standard deviation of mean of β, and the rate of noisy features (quantity of noisy features/Total number of features).

i. Both algorithms, iMWK-Means and WK-Means, tend to benefit from a reduction of similar features in the datasets.

6.3 Limitations

Clustering algorithms that perform feature weighting have some general limitations, and the two we have introduced are no different. This type of algorithms calculates the feature weights based on one feature at a time, and are by consequence unable to detect interacting features in a dataset. The information this type of features carry cannot be detected on an individual basis, the XOR problem is a good example of such. Another point is that such methods seem to excel when noise is distributed according to features and we have not performed experiments in which the noise is distributed according to the entities.

At times feature weighting algorithms may set a negligible weight to a feature, and even so the weight and by consequence the contribution of the feature to the clustering task are negligible, it still generates a much higher computational effort than

if the feature had not being used at all.

Probably, the biggest limitation of the weighting algorithms we introduced is that we did not present a clear method to find the Minkowski exponent that stably works in different datasets. Unfortunately this issue reminds as an open problem for our algorithms as it does to the WK-Means parameter (Chan et al., 2004; Huang et al., 2005, 2008).

In terms of MWK-Means only, this algorithm can be more computationally demanding than WK-Means because the Minkowski distance and Minkowski centre are indeed more difficult to calculate than the Euclidean distance and the mean of a cluster. Of course, the addition of the anomalous pattern mechanism has helped iMWK-Means on this regard.

The iKFS algorithm deals only with the removal of similar features and its parts from the principle that if two or more features are in the same cluster, they are similar and by consequence carry similar information. It is intuitive for us that clustering algorithms may or may not correctly cluster a dataset, and that a wrong clustering of features will potentially have a negative impact on the feature selection. We also have to acknowledge that although it had a good outcome, our experiment deals with only one type of data, the power spectrum density of EEG data explained in Section 5.2. Finally, the measure of the outcome was performed using only two clustering algorithms and a single index – their accuracy.

6.4 Future research

We consider the algorithms we have introduced to be rather flexible and they allow us a wide choice of future research. The most clear starting point for future research is to

develop a clear method to find the Minkowski exponent or a framework to cluster data at unknown β and labels. There are different possibilities regarding this, perhaps by analysing the behaviour of β and the shape of the datasets or in the case of the former maybe using semi-supervised learning and pair-wise roles, along the lines set by Wagstaff et al (2001), which we have applied to iK-Means in a previous publication (Amorim, 2008).

Regarding the iKFS algorithm, we intend to perform further validation tests perhaps with different tasks than clustering. It could be interesting expanding it to perform subspace feature selection.

References

Aggarwal, C.C., Hinneburg, A., Keim D. (2000), On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (Eds.) *ICDT 2001*. LNCS, Springer, vol. 1973, pp. 420-434.

Agrawal, R., Gehrke, J., Gunopulos, D., and Raghavan P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. *In Proceedings ACM SIGMOD International Conference on Management of Data*, Seattle, WA, ACM Press, pp. 94–105.

Agrawala, A. K.(1970), Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, vol 16, pp 373–379.

Aldenderfer , M. and Blashfield R. (1984), *Cluster analysis* . Newbury Park, CA : Sage Publications.

Allison, D.B., Cui, X., Page, G.P., et al. (2006). Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet*, vol 7, pp. 55–65.

Amorim, R. C. (2008), Constrained Intelligent K-Means: Improving Results with Limited Previous Knowledge, *Proceeding of Advanced Engineering Computing and Applications in Sciences*, IEEE Computer Society, pp 176-180.

Amorim, R. C., Mirkin, B., Gan J. Q. (2010), A Method for Classifying Mental Tasks in the Space of EEG Transforms. *Technical report BBKCS-10-01*, Birkbeck University of London, London.

Arthur, D., Vassilvitskii, S. (2006) How slow is the k-means method? In *SCG '06: Proceedings of the twenty-second annual symposium on computational geometry*. ACM Press.

Asuncion, A., Newman, D.J., UCI Machine Learning Repository [http://www.ics.uci. edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science, 2007. Accessed 3 April 2010.

Ball, G. and Hall, D. (1967) A clustering technique for summarizing multivariate data, *Behav. Sci.*, vol. 12, pp. 153–155.

Banerjee, A., Merugu, S., Dhillon I., Ghosh J. (2005), Clustering with Bregman divergences, *Journal of Machine Learning Research*, vol 6, pp. 1705–1749.

Bar-Hillel, A., Hertz, T., Shental, N., and Weinshall, D. (2003). Learning distance functions using equivalence relations. *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, pp. 11–18.

Basu, S., Banerjee, A., & Mooney, R. J. (2002). Semi-supervised clustering by seeding. *Proceedings of 19th International Conference on Machine Learning*, pp. 19–26.

Belacel, N., Raval, H. B., Punnen, A. P. (2007), Learning multicriteria fuzzy classification method *PROAFTN* from data, *Computers and Operations Research*, vol 34, pp. 1885–1898.

Bellman, R. E. (1957) *Dynamic Programming*. Rand Corporation, Princeton University Press.

Berkhin, P. (2006) *A survey of clustering datamining techniques*. In Jacob Kogan, Charles Nicholas, and Marc Teboulle (eds.), Grouping Multidimensional Data: Recent Advances in Clustering, pp. 25–71. Springer. 372, 520.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999), When is Nearest Neighbor Meaningful?. *Proc. 7th International Conference on Database Theory – ICDT*, LNCS 1540, pp. 217–235.

Bilenko, M., Basu, S. And Mooney, R.J. (2004), Integrating constraints and metric learning in semi-supervised clustering, *Proceedings of the 21st international conference on machine learning*, Banff, Canada, pp. 81-88

Blockeel, H., De Raedt L., and Ramon, J. (1998) *Top-down induction of clustering trees*, In: J. Shavlik, (Ed.) *Proceedings of the 15th International Conference on Machine Learning*, pp. 55-63.

Bradley, P. S., Mangasarian, O. L., and W., Street, N. (1997), Clustering via Concave Minimization, In: *Advances Neural Information Processing Systems 9*, M. C. Mozer, M. Jordan, and T. Petsche (Eds.) pp 368-374, MIT Press.

Brohee S, Van Helden J. (2006) Evaluation of clustering algorithms for protein–protein interaction networks, *BMC Bioinformatics* 7:488.

Callan, R. (2003) Artificial Intelligence. *Palgrave Macmillan*, NY, USA.

Cao, F., Liang, J., Jiang, G. (2009) An initialization method for the K-Means algorithm using neighborhood model. *Comput. Math. Appl*. 58:3, pp. 474-483.

Chakerian, G. D. and Ghandehari, M. A. (1985) The Fermat problem in Minkowski spaces. *Geometriae Dedicata*, 17, pp. 227-238.

Chan, Y., Ching, W. K., Ng M. K., and Huang J. Z. (2004), An optimization algorithm for clustering using weighted dissimilarity measures. *Pattern Recognition*. 37:5, pp. 943-952.

Chapelle, O., Scholkopf, B., and Zien, A., (2006), Semi supervised learning, The MIT Press.

Chen, Y., Rege, M., Dong, M., and Hua, J. (2008), Non-negative matrix factorization for semi-supervised data clustering, *Knowedge Information Systems*. 17:3, pp. 355-379.

Chiang, M. M. and Mirkin, B.(2010) Intelligent choice of the number of clusters in k-means clustering: An experimental study with different cluster spreads, *Journal of Classification*. 27:1, pp. 1-38.

Chiappa, S., and Bengio S. (2004), Hmm and iohmm modeling of eeg rhythms for asynchronous bci systems. *In European Symposium on Articial Neural Networks ESANN*.

Clerc, M. and Kennedy, J. (2002) The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 58-73

Dash, M., Choi, K., Scheuermann, P. and Liu, H. (2002) Feature Selection for Clustering-a Filter Solution, *Proc. Second International Conf. Data Mining*, pp. 115-122.

Davies , D. and Bouldin , D. (1979), A cluster separation measure . *IEEE Transactions on Pattern Analysis and Machine Intelligence* , 1, pp. 224–227 .

Desarbo, W. S., Carroll, J. D., Clark, L. A., Green, P. E. (1984) Synthesized clustering: A method for amalgamating clustering bases with differential weighting variables. *Psychometrika*. 49:1, pp. 57-78.

Devijver, P. A., and Kittler, J. (1982), *Pattern Recogntion: A statistical approach*. Englewood Cliffs: Prentice Hall.

Deza, M. and Deza, E. (2009) *Encyclopedia of Distances*. Springer.

Dhillon, I., Mallela, S., and Kumar, R. (2003), A divisive information-theoretic feature clustering algorithm for text classification. *Journal of Machine Learning Research*, vol 3, pp.1265-1287.

Doherty, K. A. J., Adams, R. G., and Davey, N. (2004), Non-Euclidean norms and data normalization. *Proceedings of European Symposium on Artificial Neural Networks*, pp. 181-186.

Drineas, P., Frieze, A., Kannan, R., Vempala, S., and Vinay, V. (1999) Clustering large graphs via the singular value decomposition. *Machine learning*, 56(1-3), pp 9–33.

Drongelen, W. V. (2007), *Signal Processing for Neuroscientists: An Introduction to the Analysis of Physiological Signals*, Academic Press/Elsevier, Burlington MA. USA.

Dubes, R. (1987) How Many clusters are the best? – an experiment, *Pattern Recognition*, 20 (6), pp. 645-663

Duda, R. O. and Hart, P. E. (1973) *Pattern Classification and Scene Analysis*, New York: J. Wiley & Sons.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001) *Pattern Classification*. John Wiley and Sons, Inc.. New York, 2nd Edition.

Dy, J G. (2008) Unsupervised Feature Selection. In: H. Liu and H. Motoda (Ed.) *Computational Methods of Feature Selection*, Chapman & Hall/CRC, pp. 19-39.

Dy, J. G., and Brodley, C. E. (2004), Feature selection for unsupervised learning. *Journal of Machine Learning Research*, vol 5, pp. 845–889.

Estivill-Castro V. (2002). Why so many clustering algorithms – a position paper. *SIGKDD Explorations*, 4(1):, pp. 65-75.

Everitt, B., Landau, S., and Leese, M. (2001), *Cluster analysis*, 4th edition . London : Arnold .

Fan, J., Han, M., Wang, J. (2009) Single point iterative weighted fuzzy C-means clustering algorithm for remote sensing image segmentation. *Pattern Recognition*. 42:11, pp. 2527-2540.

Fralick, S. C. (1967), Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, vol. 13, pp. 57–64.

Francois, D., Wertz, V., and Verleysen M. (2007), The concentration of fractional distances, *IEEE Transactions on Knowledge and Data Engineering*, 19(7), pp. 873-886.

Friedman, J. H., Meulman, J. J. (2004) Clustering objects on subsets of attributes. *Journal of the Royal Statistical Society B*, vol 66, 4, pp. 815-849.

Frigui, H., Nasraoui, O. (2004) Unsupervised learning of prototypes and attribute weights, *Pattern Recognition*. 37, pp. 567-581.

Gan, J.Q. (2006), Self-adapting bci based on unsupervised learning. *In 3rd International Workshop on Brain-Computer Interfaces*, Graz, Austria, pp. 50–51.

Geng, T., Gan, J. Q., Dyson, M., Tui, C. S. L. and Sepulveda, F. (2008), A novel design of 4-class BCI using two binary classifiers and parallel mental tasks, *Journal of Computational Intelligence and Neuroscience*, vol 2008, DOI: 10.1155/2008/437306.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Kluwer Academic Publishers.

Green, P. E., Carmone, J., Kim, J. (1990) A preliminary study of optimal variable weighting in k-means clustering. *Journal of Classification*, 7:2, pp. 271-285.

Guyon, I., And Elisseeff, A. (2003), An introduction to variable and feature selection. *J. Mach. Learn. Res*. vol 3, pp.1157–1182.

Hand, D, Mannila, H. And Smyth, P., (2001), *Principles of Data Mining*, MIT Press.

Hansen, P. and Mladenovic, N. (2001) Variable neighborhood search: Principles and applications. *Europ. J. Oper. Res*. 130, pp. 449–467.

Hartigan, J. A. (1975) *Clustering Algorithms*, John Wiley and Sons, Inc., New York, NY. USA.

Hartigan, J. A., Wong, M. A. (1979) Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society, Series C (Applied Statistics)* 28(1), pp 100–108.

Holzinger, K. J. and Harman, H. H. (1941) *Factor Analysis*, Chicago: University of Chicago Press, USA.

Hong, Y., Kwong, S., Chang, Y., Ren, Q. (2008) Unsupervised feature selection using clustering ensembles and population based incremental learning algorithm. *Pattern Recognition.* 41:9, pp. 2742-2756.

Huang J. Z., Ng M. K., Rong, H., and Li, Z. (2005), Automated variable weighting in k-means type clustering. *IEEE Transactions on Pattern Analysis and Machine Learning*, 27:5, pp. 657-668.

Huang, J. Z., Xu, J., Ng, M. And Ye, Y. (2008) Weighting Method for Feature Selection in K-Means. In: H. Liu and H. Motoda (Ed.) *Computational Methods of Feature Selection*, Chapman & Hall/CRC, pp 193-209.

Huber, R., Treyer, V., Borbe, A. A., Schuderer, J., Gottselig, J. M., Landol, H. P., Werth, E., Berthold, T., Kuster, N., Buck, A., Achermann, P. (2002), Electromagnetic fields, such as those from mobile phones alter regional cerebral blood flow and sleep and awaking EEG, *Journal of Sleep Research*, vol. 11, no. 4, pp. 289-295

Jain, A. K. (2010), Data Clustering: 50 Years Beyond K-Means, *Pattern Recognition Letters*, vol 31 pp 651-666.

Jain, A. K. And Dubes, R. C. (1988) *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA.

Kaufman, L. and Rousseeuw, P. (1990) *Finding Groups in Data: An Introduction to Cluster Analysis*, New York: J. Wiley & Son.

Kennedy, J. and Eberhart, R. C. (1995) Particle swarm optimization, *Proc. IEEE Int. Conf. Neural Networks*, Perth, Australia, pp. 1942–1948.

Kim, J., Shim, K. H., and Choi S. (2007) Soft geodesic kernel *k*-means, in *Proc. 32nd IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pp. 429-432.

King, B. (1967), Step-wise clustering procedures, *J. Am. Statistical Assoc.*, pp.86-101.

Kivinen, J., Warmuth, M. K., Hassibi B. (2006), The p-Norm generalization of the LMS algorithm for adaptive filtering, *IEEE Transactions on Signal Processing*, 54(5), pp. 1782-1793.

Klein, D., Kamvar, S. D., & Manning, C. (2002). From instancelevel constraints to space-level constraints: Making the most of prior knowledge in data clustering. *Proceedings of the The Nineteenth International Conference on Machine Learning (ICML-2002)*, pp. 307–314.

Klitzing, L. V. (1995), Low-frequency pulsed electromagnetic fields influence EEG of man, *Physica Med*, vol. 11, pp. 77-90

Koggalage, R., Halgamuge, S.(2004) Reducing the Number of Training Samples for Fast Support Vector Machine Classification. *In Neural Information Processing*. 2:3, pp. 57-65

Kononenko, I. (1994), Estimating attributes: Analysis and extensions of Relief. In *Proceedings of the European Conference on Machine Learning*.

Kriegel, H. P., Kröger, P., Zimek, A. (2009), Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering, *ACM Transactions on Knowledge Discovery from Data* (New York, NY: ACM), 3 (1), pp. 1–58.

Kumar, N. and Kummamuru, K. (2007), Semi-supervised clustering with metric learning using relative comparisons. *IEEE Trans. on Knowledge and Data engineering*, 20(4), pp. 496-503.

Lawrence, V. S. (2006) Facility location under uncertainty: A review. *IIE Transactions*, 38(7), pp. 537–554.

Lee, F., Scherer, R., Leeb, R., Neuper, C., Bischof, H., Pfurtscheller, G. (2005), A Comparative analysis of multi-class EEG classification for brain computer interface, *Proceedings of the 10th Computer Vision Winter Workshop*, pp.195-204

Li, H., and Motoda, H. (2008) Less Is More. In: H. Liu and H. Motoda (Ed.) *Computational Methods of Feature Selection*, Chapman & Hall/CRC, pp. 3-17.

MacQueen, J. (1967) Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pp. 281–297.

Makarenkov, V., Legendre, P. (2001), Optimal variable weighting for ultrametric and additive trees and K-Means partitioning, *Journal of Classification*. Vol 18, pp. 245-271.

Manning, C. D., Raghavan P., and Schutze, H. (2008) *Introduction to Information Retrieval*. Cambridge University Press, UK.

Marill, T. and Green, D. M. (1963) On the Effectiveness of receptors in recognition systems. *IEEE Transactions on Information Theory*, vol 9, pp. 11-17.

Maulik, U. and Bandyopadhyay, S. (2000) Genetic algorithm-based clustering technique. *Pattern Recognition*, 33, pp. 1455–1465.

McLachlan, G.J., Khan, N. (2004) On a resampling approach for tests on the number of clusters with mixture model-based clustering of tissue samples, *Journal of Multivariate Analysis*. Vol 90, 1005.

Millan, J. R. and Mourino, J. (2003) Asynchronous BCI and local neural classifiers: An overview of the Adaptive Brain Interface project. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Special Issue on Brain-Computer Interface Technology.

Milligan, G. W. And Cooper, M. C. (1988) A study of standardization of the variables in cluster analysis, *Jornal of Classification*, 5, pp. 181-204.

Mingoti, S. A., Lima, J. O. (2006) Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research* 174, pp1742-1759.

Mirkin, B. (1996), *Mathematical classification and clustering*. Dordrecht: Kluwer Academic Pub.

Mirkin, B. (2005) *Clustering for Data Mining: A Data Discovery Approach*, Boca Raton Fl., Chapman and Hall/CRC.

Mitra, P., Murthy,C., And Pal, S. (2002), Unsupervised feature selection using feature similarity. *IEEE Trans. Pattern Anal. Mach. Intell*. 24, 3, pp. 301–312.

Mladenovic, N. and Hansen, P. (1997) Variable neighborhood search. *Computers and Operations Research*, vol. 24, pp. 1097-1100.

Modha, D.S., Spangler, W.S. (2003) Feature weighting in K-Means clustering, *Machine Learning*. 52, pp. 217-237.

Mojena, R. (1977) Hierarchical grouping methods and stopping rules: an evaluation, *The Computer Journal*, 20:4, pp. 359-363.

Monti, S., Tamayo, P., Mesirov, J. and Golub, T. (2003) Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data, *Machine Learning*. Vol 52, pp. 91-118.

Mucherino, A., Papajorgji, P., Pardalos, P.M.(2009) *Data Mining in Agriculture*, Springer.

Murtagh, F. (2009), The remarkable simplicity of very high dimensional data: application to model-based clustering. *Journal of Classication*, vol. 26, pp. 249-277.

Nascimento, S. (2005) *Fuzzy Clustering Via Proportional Membership Model*. In: Frontiers in Artificial Intelligence and Applications, vol. 119, IOS Press, Amsterdam.

Netlab Neural Network software, http://www.ncrg.aston.ac.uk/netlab/index.php, accessed 3 April 2010.

Ortolani, O., Conti, A., Di Filippo, A., Adembri, C., Moraldi, E., Evangelisti, A., Maggini, M., Roberts, S. J. (2002), EEG signal processing in anaesthesia: Use of neural network technique for monitoring depth of anaesthesia, *British Journal of Anaesthesia* , vol. 88, no. 5, pp. 644-648.

Pal, S.K., Mitra, P. (2004) *Pattern Recognition Algorithms for Data Mining*, CRC Press, Boca Raton, Florida

Pedrycz, W (2005) *Knowledge-Based Clustering: From Data to Information Granules*. Hoboken, NJ: Wiley.

Pelleg, D., Moore, A. (2000) Xmeans: Extending kmeans with efficient estimation of the number of clusters, *Proc. 17th International Conference on Machine Learning*, pp. 727-734.

Perlibakas, V. (2004) Distance measures for PCA-based face recognition, *Pattern Recognit. Lett.*, vol. 25, no. 6, pp. 711-724.

Pollard, K.S. and Laan, V. M. J. (2002) A method to identify significant clusters in gene expression data, *U.C. Berkeley Division of Biostatistics Working Paper Series*, 107.

Pudil, P., Novovicova, J., and Kittler, J. (1994), Floating Search Methods in Feature Selection, *Pattern Recognition Letters*, vol. 15, pp. 1119-1125.

Raftery, A. E., Dean, N. (2006) Variable selection for Model-based clustering. *Journal of the American Statistical Association*, 101, pp. 168-178.

Rand , W. (1971), Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, vol 66, pp.846–850 .

ReVelle, C.S. and Eiselt, H.A. (2005) Location analysis: A synthesis and survey. *Euro. J. Oper. Res.*, 165(1), pp. 1–19.

Romesburg, C. (2004), *Cluster Analysis For Researchers*. Morrisville, NC: Lulu Press.

Rudin, B. (2009), The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list, *Journal of Machine Learning Research*, vol 10, pp. 2233-2271.

Sanei, S., Chambers, J. A. (2007), *EEG Signal Processing*. UK: WileyBlackwell.

Scudder, H. J. (1965), Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, vol 11, pp.363–371.

Sneath, P.H.A. and Sokal, R. R. (1973) *Numerical Taxonomy*, San Francisco: W.H. Freeman.USA.

Smith, S. P., And Jain, A. K. (1984), Testing for uniformity in Multidimensional data. *IEEE Trans. on pattern analysis and machine intelligence*, 6(1), pp. 73–81.

Stanforth, R., Kolossov, E. and Mirkin, B. (2007), A Measure of Domain of Applicability for QSAR Modelling Based on Intelligent K-Means Clustering. *QSAR. Comb. Sci*., vol s26, pp. 837–844.

Steinley, D., and Brusco, M. (2007), Initializing K-Means Batch Clustering: A Critical Evaluation of Several Techniques, *Journal of Classification*, 22, pp. 221-250.

Stracuzzi, D. J. (2008) Randomized Feature Selection. In: H. Liu and H. Motoda (Ed.) *Computational Methods of Feature Selection*, Chapman & Hall/CRC, pp. 41-62.

Strehl, A., Ghosh, J., Mooney, R. J., Impact of similarity measures on web-page clustering. *Proc. of AAAI Workshop on AI for Web Search*, 2000.

Szeliski, R. (2010) *Computer Vision: Algorithms and Applications*. Springer.

Tibshirani, R., Walther G. And Hastie, T. (2001) Estimating the number of clusters in a dataset via the Gap statistics, *Journal of the Royal Statistical Society B*. Vol 63, pp. 411-423.

Tsai, C.Y., Chiu, C.C. (2008) Developing a feature weight adjustment mechanism for a K-Means clustering algorithm, *Computational Statistics and Data Analysis*. 52, pp. 4658-4672.

Tsiptsis, K., Chorianopolous, A. (2009) *Data Mining Techniques in CRM: Inside Customer Segmentation.* UK: John Wiley & Sons Ltd..

Tsui, C., Gan, J. Q., Roberts, S. (2009), A self-paced brain-computer interface for controlling a robot simulator: an online event labelling paradigm and an extended Kalman filter based algorithm for online training, *Medical and Biological Engineering and Computing*, vol. 47, pp 257-265.

Ungureanu, M., Bigan, C., Strungaru, R., Lazarescu, V. (2004), Independent component analysis applied in biomedical signal processing, *Measurement Science Review*, vol. 4, Section 2, pp. 1-8

Wagstaff, K. L., Basu, S., and Davidson, I. (2006), When is Constrained Clustering Benefical and why?, *The 21st National Conference on Artificial Intelligence and the 18th Innovative Applications of Artificial Intelligence Conference*, AAAI Press, Boston USA.

Wagstaff, K. L., Cardie, C., Rogers, S., and Schroedl, S. (2001), Constrained K-means Clustering with Background Knowledge, *Proceedings of the 18th International Conference on Machine Learning*, pp. 577-584

Ward Jr, J.H. (1963) Hierarchical grouping to optimize an objective function, *Journal of the American Statistical Association*, vol 58, pp. 236-244.

Webb, A. (2002) *Statistical Pattern Recognition*, John Wiley and Sons, Inc., UK.

Whitney, A. W. (1971) A direct method of nonparametric measurement selection. *IEEE Transactions on Computers*, vol 20, pp. 1100-1103.

Witten, I. H. and Frank E. (2005) *Data Mining: Practical Machine Learning Tools and Techniques*. San Francisco: Morgan Kaufmann, 2nd Edition.

Wong, P. C., Foote, H., Adams, D., Cowley, W., Leung, L. R., Thomas, J. (2005) *Visualizing Data Streams*. In: Kovolerchuk, B. And Schwing, J., ed. Visual and Spatial Analysis – Advances in Data Mining, Reasoning, and Problem Solving. Springer.

Xing, E. P., Karp, R. M. (2001), Cliff: Clustering of high-dimensional microarray data via iterative feature filtering using normalised cuts. In *9th International Conference on Intelligence Systems for Molecular Biology*.

Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S. (2003). Distance metric learning, with application to clustering with sideinformation. *Advances in Neural Information Processing Systems*, vol 15, pp. 505–512.

Xu, R., and Wunsch II, D. C. (2005) Survey of clustering algorithms. *IEEE Trans. on Neural Networks*, 16(3), pp 645–678.

Xu, R., and Wunsch II, D. C. (2010), *Clustering*, John Wiley & Sons, 2010

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

Zainuddin, Z., Lye, W. K. (2010), Improving RBF Networks Classification Performance by using K-Harmonic Means, *World Academy of Science, Engineering and Technology*, vol 62, pp. 983-986.

Zeidat, N. M., Eick, C. F. (2004), K-medoid-style clustering algorithms for supervised summary generation. *IC-AI*, pp. 932-938.

Zhong, L., Jinsha, Y., Weihua, Z. (2009), Fuzzy C-Mean Algorithm with Morphology Similarity Distance, *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. 3, pp. 90-94.

Zhu, X., Rogers, T., Qian, R., and Kalish, C., (2007) Humans perform semi-supervised classification too. *22nd AAAI conference on Artificial Intelligence*, AAAI Press California, pp 864-870.

Zilberstein, S. (1996) Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3).

Appendix A: Results of general experiments per dataset

The tables in this section show the results obtained with all K-Mean based algorithms in all Gaussian models, rather than a summary of the results, as shown in Chapter 3. A brief description of each Gaussian model can be found in the tables' captions. Chapter 3 presents an extended description in Section 3.4.

Dataset	Algorithm	Exponent at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
GM 1	K-Means	2	-	36.2	5.43	51.4
	WK-Means	2	1.6	58.0	11.08	78.0
	WMK-Means	1.6	1.6	61.3	9.83	86.2
	iK-Means	2	-	-	-	33.6
	iWK-Means	2	2.1	-	-	65.8
	iWMK-Means	2.9	2.9	-	-	67.0
GM 2	K-Means	2	-	37.2	2.23	45.2
	WK-Means	2	1.5	57.4	9.12	74.8
	WMK-Means	1.5	1.5	61.05	8.89	74.0
	iK-Means	2	-	-	-	37.0
	iWK-Means	2	1.5	-	-	69.6
	iWMK-Means	1.4	1.4	-	-	69.4
GM 3	K-Means	2	-	38.1	1.88	42.6
	WK-Means	2	1.5	46.9	5.58	57.6
	WMK-Means	1.6	1.6	47.2	7.07	63.2
	iK-Means	2	-	-	-	39.0
	iWK-Means	2	1.6	-	-	51.2
	iWMK-Means	1.9	1.9	-	-	48.6
GM 4	K-Means	2	-	48.5	5.21	57.8
	WK-Means	2	1.4	65.6	9.11	90.6
	WMK-Means	1.4	1.4	71.8	10.64	90.2

	iK-Means	2	-	-	-	50.4
	iWK-Means	2	1.4	-	-	73.0
	iWMK-Means	1.3	1.3	-	-	74.8
GM 5	K-Means	2	-	32.3	3.23	41.8
	WK-Means	2	1.4	54.7	11.23	89.4
	WMK-Means	1.5	1.5	59.7	12.41	88.0
	iK-Means	2	-	-	-	28.6
	iWK-Means	2	1.7	-	-	53.4
	iWMK-Means	1.6	1.6	-	-	87.6
GM 6	K-Means	2	-	41.7	4.65	51.2
	WK-Means	2	1.6	56.9	7.91	72.6
	WMK-Means	1.6	1.6	58.1	6.55	71.6
	iK-Means	2	-	-	-	40.6
	iWK-Means	2	1.7	-	-	65.2
	iWMK-Means	1.6	1.6	-	-	66.0
GM 7	K-Means	2	-	39.2	5.4	53.8
	WK-Means	2	1.6	59.5	8.87	79.4
	WMK-Means	1.5	1.5	63.5	8.76	85.4
	iK-Means	2	-	-	-	35.2
	iWK-Means	2	4.7	-	-	75.0
	iWMK-Means	2	2	-	-	69.6
GM 8	K-Means	2	-	34.3	2.39	42.4
	WK-Means	2	1.4	54.7	9.48	77.2
	WMK-Means	1.6	1.6	61.7	12.02	86.2
	iK-Means	2	-	-	-	32.4
	iWK-Means	2	1.8	-	-	85.4
	iWMK-Means	1.6	1.6	-	-	85.4
GM 9	K-Means	2	-	32.5	1.79	38.0
	WK-Means	2	1.5	50.7	10.78	85.2
	WMK-Means	1.6	1.6	54.7	10.83	83.4
	iK-Means	2	-	-	-	32.8
	iWK-Means	2	3.0	-	-	49.8
	iWMK-Means	1.4	1.4	-	-	57.0
GM 10	K-Means	2	-	45.0	2.27	52.0
	WK-Means	2	1.5	61.2	9.50	80.8

	WMK-Means	1.5	1.5	63.6	7.72	77.0
	iK-Means	2	-	-	-	47.4
	iWK-Means	2	1.9	-	-	72.8
	iWMK-Means	2.2	2.2	-	-	77.4

Table A.1: Accuracy levels at the considered versions of K-Means for dataset (vii) – 500 entities, 12 clusters, eight features including 2 extra noise features.

Dataset	Algorithm	Exponent at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
GM 1	K-Means	2	-	46.31	3.98	55.2
	WK-Means	2	3.9	64.69	9.22	94.4
	WMK-Means	1.3	1.3	74.11	10.27	93.8
	iK-Means	2	-	-	-	49.8
	iWK-Means	2	1.3	-	-	72.8
	iWMK-Means	1.4	1.4	-	-	94.49
GM 2	K-Means	2	-	50.82	8.0	64.40
	WK-Means	2	2.4	66.97	8.97	81.0
	WMK-Means	1.4	1.4	75.88	7.28	91.8
	iK-Means	2	-	-	-	58.8
	iWK-Means	2	4.6	-	-	77.6
	iWMK-Means	1.6	1.6	-	-	91.6
GM 3	K-Means	2	-	32.55	5.84	57.4
	WK-Means	2	4.8	85.08	12.4	98.4
	WMK-Means	1.4	1.4	87.30	10.23	97.8

	iK-Means	2	-	-	-	33.2
	iWK-Means	2	2	-	-	85.6
	iWMK-Means	1.7	1.7	-	-	97.6
GM 4	K-Means	2	-	45.06	6.1	59.4
	WK-Means	2	1.3	64.15	8.5	77.8
	WMK-Means	1.5	1.5	72.48	7.95	87.0
	iK-Means	2	-	-	-	59.4
	iWK-Means	2	4.1	-	-	72.6
	iWMK-Means	1.3	1.3	-	-	74.0
GM 5	K-Means	2	-	58.79	9.13	77.4
	WK-Means	2	4.8	81.32	7.43	97.6
	WMK-Means	1.4	1.4	84.31	7.28	96.4
	iK-Means	2	-	-	-	78.0
	iWK-Means	2	3.9	-	-	95.2
	iWMK-Means	2	2	-	-	86.4

Table A.2: Accuracy levels at the considered versions of K-Means for the 5 GM datasets with 500 entities, 15 features and 5 clusters with an additional 10 random-noise features.

Dataset	Algorithm	Exponent at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
GM 1	K-Means	2	-	23.09	3.97	34.7
	WK-Means	2	5	60.28	7.06	76.9
	WMK-Means	1.3	1.3	72.8	7.24	85.8
	iK-Means	2	-	-	-	28.8
	iWK-Means	2	3.2	-	-	46.0
	iWMK-Means		1.7	-	-	59.5
GM 2	K-Means	2	-	21.49	2.98	28.7
	WK-Means	2	4.4	49.11	7.87	77.2
	WMK-Means	1.3	1.3	69.29	7.13	84.0
	iK-Means	2	-	-	-	17.0
	iWK-Means	2	2.7	-	-	45.5
	iWMK-Means		1.3	-	-	51.4
GM 3	K-Means	2	-	18.37	2.16	25.1
	WK-Means	2	4.7	45.92	7.15	60.8
	WMK-Means	1.4	1.4	64.54	8.2	82.0
	iK-Means	2	-	-	-	15.8
	iWK-Means	2	4.8	-	-	36.3
	iWMK-Means		1.3	-	-	56.8
GM 4	K-Means	2	-	21.26	2.88	28.7
	WK-Means	2	4.9	52.93	6.87	71.5
	WMK-Means	1.3	1.3	66.37	8.57	83.7
	iK-Means	2	-	-	-	27.0

	iWK-Means	2	2.7	-	-	53.2
	iWMK-Means		1.6	-	-	54.0
GM 5	K-Means	2	-	19.5	2.15	25.8
	WK-Means	2	4.8	52.03	7.28	70.4
	WMK-Means	1.3	1.3	69.16	8.56	85.2
	iK-Means	2	-	-	-	21.0
	iWK-Means	2	1.0	-	-	15.8
	iWMK-Means	2	1.5	-	-	52.7

Table A.3: Accuracy levels at the considered versions of K-Means for the 5 GM datasets with 1000 entities, 25 features and 12 clusters with an additional 25 random-noise features.

Dataset	Algorithm	Exponent at		Accuracy, %		
		Distance	Weight	Mean	Std	Max
GM 1	K-Means	2	-	92.31	7.98	1
	WK-Means	2	5	79.15	8.13	93.0
	WMK-Means	1.2	1.2	83.99	10.41	99.9
	iK-Means	2	-	-	-	88.6
	iWK-Means	2	1.2	-	-	24.1
	iWMK-Means	1.8	1.8	-	-	84.6
GM 2	K-Means	2	-	92.48	6.37	1
	WK-Means	2	4.5	81.94	7.15	94.2
	WMK-Means	1.3	1.3	84.29	6.76	93.5
	iK-Means	2	-	-	-	89.3
	iWK-Means	2	4.2	-	-	78.4

	iWMK-Means	2	2	-	-	69.8
GM 3	K-Means	2	-	91.41	6.72	1
	WK-Means	2	4	80.2	7.7	92.9
	WMK-Means	1.2	1.2	84.56	7.1	100
	iK-Means	2	-	-	-	87.9
	iWK-Means	2	1.0	-	-	16.2
	iWMK-Means	2.6	2.6	-	-	61.7
GM 4	K-Means	2	-	93.95	6.34	0.99
	WK-Means	2	2.9	81.02	7.55	92.9
	WMK-Means	1.2	1.2	86.5	8.88	100
	iK-Means	2	-	-	-	99.9
	iWK-Means	2	1.2	-	-	22.8
	iWMK-Means	1.8	1.8	-	-	61.6
GM 5	K-Means	2	-	93.81	6.76	1
	WK-Means	2	4.5	79.81	7.41	92.6
	WMK-Means	1.2	1.2	86.75	7.77	100
	iK-Means	2	-	-	-	76.9
	iWK-Means	2	1.6	-	-	45.3
	iWMK-Means	1.3	1.3	-	-	76.1

Table A.4: Accuracy levels at the considered versions of K-Means for the 5 GM datasets with 1000 entities, 50 features and 12 clusters with an additional 25 random-noise features.

Appendix B: Learning the exponent: Results of experiments per dataset

The tables in this section show the results obtained with WK-Means and iMWK-Means while attempting to learn their parameter using semi-supervised learning. Chapter 4 provides more details in such experiments and a summary of them. A brief description of each Gaussian model can be found in the tables' captions, while Chapter 3 presents an extended description in Section 3.4.

(i) Experiments with the weighted K-Means

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.99	0.06	1.7	1.6	55.59	1.88	58.52
2	1.67	0.04	1.4	1.5	54.95	2.41	59.24
3	2.28	0.10	1.6	1.5	44.27	1.79	47.45
4	1.69	0.06	1.4	1.4	62.9	1.83	66.4
5	1.76	0.06	1.5	1.4	51.5	3.56	56.33
6	2.01	0.08	1.4	1.6	55.49	1.28	57.73
7	2.06	0.08	1.5	1.6	57.43	1.39	60.13
8	1.65	0.04	1.5	1.4	53.03	2.07	56.78
9	1.55	0.04	1.4	1.5	48.8	2.10	52.94
10	2.27	0.09	1.7	1.5	57.37	2.14	61.26

Table B.1: Semi-supervised estimations of β in WK-Means on Gaussian models with noise, dataset (vii) - 500x8, including 2 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.51	0.04	1.4	3.9	63.79	0.93	66.01
2	1.53	0.02	1.6	2.4	66.18	0.96	69.32
3	1.46	0.01	1.4	4.8	76.05	1.89	81.14
4	1.44	0.01	1.4	1.3	63.19	1.17	65.48
5	1.49	0.02	1.4	4.8	77.23	1.51	79.76

Table B.2: Semi-supervised estimations of β in WK-Means on Gaussian models with noise, dataset (viii) - 500x25, including 10 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.36	0.09	1.3	5.0	52.28	1.13	54.51
2	1.36	0.06	1.4	4.4	46.43	1.45	50.29
3	1.38	0.09	1.3	4.7	41.3	0.91	44.27
4	1.41	0.06	1.4	4.9	46.62	0.88	48.29
5	1.38	0.09	1.3	4.8	45.90	1.55	48.62

Table B.3: Semi-supervised estimations of β in WK-Means on Gaussian models with noise dataset (ix) - 1000x50, including 50 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.78	0.06	1.4	5.0	74.6	1.52	81.12
2	1.96	0.07	1.8	4.5	76.83	1.39	80.19
3	1.89	0.06	1.6	4.0	75.48	1.19	78.31
4	1.54	0.02	1.4	2.9	75.16	0.94	77.76
5	1.54	0.03	1.4	4.5	73.72	1.13	77.34

Table B.4: Semi-supervised estimations of β in WK-Means on Gaussian models with noise, dataset (x) - 1000x75, including 25 extra noise features.

(ii) Experiments with the intelligent Minkowski Weighted K-Means

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.919	0.06	1.4	2.9	56.82	6.77	65.6
2	1.831	0.06	1.4	1.4	60.33	8.33	69.4
3	1.972	0.07	1.4	1.9	42.35	4.42	48.60
4	1.74	0.04	1.4	1.3	64.7	7.15	74.80
5	1.74	0.05	1.5	1.6	42.42	14.63	87.60
6	1.868	0.06	1.6	1.6	55.33	7.75	66.0
7	1.959	0.05	1.5	2.0	57.72	11.1	69.6
8	1.609	0.04	1.4	1.6	68.13	15.13	85.40
9	1.676	0.05	1.4	1.4	44.71	10.43	57.0
10	1.816	0.05	1.5	2.2	65.52	9.26	77.4

Table B.5: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise, dataset (vii) - 500x8, including 2 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.584	0.03	1.4	1.4	72.41	13.7	99.4
2	1.629	0.04	1.4	1.6	73.24	13.85	91.6
3	1.538	0.03	1.4	1.7	88.94	10.92	97.6
4	1.68	0.04	1.4	1.3	58.94	10.66	74.0
5	1.664	0.04	1.4	2.0	79.61	8.33	86.4

Table B.6: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise, dataset (viii) - 500x25, including 10 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	1.571	0.02	1.4	1.7	45.02	8.34	59.5
2	1.547	0.02	1.4	1.3	40.29	5.62	51.4
3	1.566	0.02	1.5	1.3	44.85	8.1	56.8
4	1.579	0.02	1.5	1.6	43.39	9.51	54.0
5	1.587	0.02	1.5	1.5	46.13	6.05	52.7

Table B.7: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise dataset (ix) - 1000x50, including 50 extra noise features.

Dataset	Exponent β at weight				% accuracy		
	Mean	Std of Mean	Modal	Optimal	Mean	Std of Mean	Max
1	2.105	0.06	1.4	1.8	47.85	14.25	84.60
2	1.887	0.05	1.3	2.0	54.19	8.19	69.80
3	1.875	0.06	1.3	2.6	46.96	9.17	61.7
4	1.791	0.05	1.4	1.8	53.67	6.87	61.6
5	1.978	0.06	1.5	1.3	52.27	12.34	76.1

Table B.8: Semi-supervised estimations of β in iMWK-Means on Gaussian models with noise, dataset (x) - 1000x75, including 25 extra noise features.