Minkowski Metric, Feature Weighting and Anomalous Cluster Initializing in K-Means Clustering

Renato Cordeiro de Amorim[1] and Boris Mirkin[1,2] (Corresponding author, e-mail: mirkin@dcs.bbk.ac.uk, tel. 44 207 631 6746 (London), 7 495 316 4641 (Moscow) fax 44 207 6316727)

[1]Department of Computer Science and Information Systems, Birkbeck University of London, Malet Street, London WC1E 7HX, UK
[2]Department of Data Analysis and Machine Intelligence, National Research University Higher School of Economics, Moscow, RF
E-mail: {renato, mirkin}@dcs.bbk.ac.uk

Abstract

This paper represents another step in overcoming a drawback of K-Means, its lack of defense against noisy features, by using feature weights in the criterion. The Weighted K-Means method by Huang et al. is extended to the corresponding Minkowski metric for measuring distances. Under Minkowski metric the feature weights become intuitively appealing feature rescaling factors in a conventional K-Means criterion. To see how this can be used in addressing another issue of K-Means, the initial setting, a method to initialize K-Means with anomalous clusters is adapted. The Minkowski metric based method is experimentally validated on datasets from the UCI Machine Learning Repository and generated sets of Gaussian clusters, both as they are and with additional uniform random noise features, and appears to be competitive in comparison with other K-Means based feature weighting algorithms.

Key-words: k-means; Minkowski metric; feature weights; noise features; anomalous cluster

1. Introduction

Clustering is a major data analysis tool used in such domains as marketing research, data mining, bioinformatics, image processing and pattern recognition [1 – 5]. K-Means is arguably the most popular clustering algorithm. It is intuitive and fast. Yet K-Means has shortcomings as well, such as:

(a) no defense against irrelevant, or noise, features;
(b) no recommendation on the initial location of cluster centroids;
(c) no criterion for selecting the number of clusters.

The main concern of this paper is regarding the issue (a), although attention is given to (b) as well. A very successful approach taken by Huang and his collaborators [5 – 7] serves as the starting point. They modify criterion of K-Means to that of their Weighted K-Means (WK-Means, for short), see later in (3), to include powers of unknown weights w_v of the variables v, $v=1, \ldots M$. This is done in the manner of the popular c-means fuzzy clustering criterion [8] except for the fact that in c-means, the weights are assigned to entities rather than features. In both formulations the weights are supposed to be non-negative and sum to unity, which alongside with the power parameters, lead to closed-form formulas for weight updating computations. It appears, the feature weights reflect within cluster feature variances: the smaller the variance, the larger the weight. Therefore, features uniformly distributed across clusters get a smaller weight while those concentrating around centroids get a larger weight [5], which is in line with intuition and works well in experiments. However, the K-Means criterion modified by Huang et al. (see later in (3)) loses its straightforward relationship between scales of the feature values and feature weights which has been underlying much of the previous work on feature weighting (for a review, see [9]). Therefore, the Euclidean metric in the criterion is in this paper extended to Minkowski's metric so that the Minkowski's exponent coincides with the exponent that is assigned to the feature weights. It appears such a modification of the criterion, further-on referred to as Minkowski metric Weighted K-Means (MWK-Means, for short), works well both theoretically and practically. On the former side, it leads to an alternating minimization algorithm similar to that by Huang et al. [5], with an added search for Minkowski centers as a process of minimization of a convex function. On the latter side, MWK-Means outperforms both K-Means and WK-Means in most experiments reported below.

To address the issues of the number of clusters (b) and initialization (c) in the context of MWK-Means criterion, a version of anomalous clustering [3] is used to initialize K-Means. In the context of the classical squared Euclidean distance K-Means criterion, this is a viable alternative to the conventional multiple runs of K-Means starting from random centroids and using popular criteria for choosing the "right number" of clusters, as experimentally demonstrated by Chiang and Mirkin [10]. In the context of weighted features, the strategy of multiple runs is not applicable to Weighted K-Means by Huang et al. [5] at different weight powers because the values of the Weighted K-Means criterion are incomparable then. Therefore, the rule for choosing the best solution at the minimum criterion value loses its ground. In contrast, the anomalous clustering strategy is applicable at all the versions of K-Means under consideration so that the results can be compared across all of them. The experiments on pre-labeled datasets reported in this paper show that using the anomalous clusters for initialization of the weighted versions of K-Means works well on real data sets and remains competitive on synthetic, generated, data.

The remainder is structured as follows. Section 2 reviews background and related work and sets goals for this paper. Section 3 introduces all versions of K-Means under consideration: the generic, so-called Batch, K-Means; the anomalous clustering procedure; the WK-Means by Huang et al; and the proposed Minkowski metric Weighted K-Means, MWK-Means. Section 4 describes experimental results with respect to the three following issues: (i) comparing proposed Minkowski metric based procedures with those proposed earlier; (ii) exploring the effects of increasing the numbers of noisy features and/or features in total, and (iii) exploring the possibility of deriving the value of Minkowski exponent in a semi-supervised manner. The experiments are conducted over several popular datasets from the UCI Machine Learning Repository [11] and on generated Gaussian cluster datasets. Section 5 concludes the paper.

2. Background and related work

In a recent review, Jain [12] formulates several issues of clustering. Four of them,

(i) Feature selection
(ii) Data normalization
(iii) Distance definition
(iv) How many clusters

are relevant to the material of this paper so that recent developments related to K-Means will be discussed in this section, as well as problems remaining to be addressed.

2.1. Feature weighting and different metrics in K-Means and related methods

The issue (i) of feature selection in clustering currently is considered either within the framework of subspace clustering (see, for a review, [13]) or within the framework of feature weighting (see, for a review, [5]). This paper belongs to the latter. Since feature weighting is part of the definition of the distance or similarity metric utilized in a clustering algorithm, this inadvertently brings us to issue (iii) of distance definition. An overwhelming majority of papers use the Euclidean squared distance $d(x,y)=(x-y)^T(x-y)=\Sigma_i(x_i-y_i)^2$ between multidimensional points x, y represented by n-dimensional column-vectors where ^{T} denotes transpose. There are three main lines for extending this measure: (1) an extension of the inner product in the distance definition; (2) nonlinear weights, and (3) other distances. These will be discussed in turn.

(1) Extension of the inner product in the distance definition involves a positive semidefinite weight matrix W so that $d(x,y)=(x-y)^TW(x-y)=\Sigma_{i,j} w_{ij}(x_i-y_i)(x_j-y_j)$, which is equivalent to using linear combinations of the original features (see, for example, [14, 15]). In a special case when W is diagonal, this amounts to $d(x,y)=(x-y)^TW(x-y)=\Sigma_i w_i(x_i-y_i)^2$, that is, using square roots $\sqrt{w_i}$ as feature weights that also are feature rescaling factors, thus relating issues (i), (ii) and (iii) above together. To choose the weights in the distance, the authors use either of two different options. One of them is to change the criterion for clustering. To accommodate the linear weights, Modha and Spangler [16] drastically change the K-Means criterion by dividing its extension, "the within-cluster dispersion" over an index expressing "the between-cluster dispersion." Another stance is taken by Tsai and Chiu [17], who do not change the criterion but rather introduce a heuristic iterative feature weights adjustment procedure. The other option involves additional information of the cluster structure. Such additional information is usually shaped as lists of pairs of entities that are to be either in the same cluster or not. Xing et al. [14] and Bilenko et al. [15] modify the K-Means criterion by adding items penalizing for breaking the constraints imposed by the lists of pairs to be put together or separately.

(2) Non-linear weights

Non-linear weights considered in the literature so far have been just weight powers so that the distance is expressed as $d(x,y)=(x-y)^T W^\beta (x-y)=\Sigma_i w_i^\beta (x_i-y_i)^2$ where W is a diagonal feature weight matrix. Probably, Makarenkov and Legendre [9] have been the first to use this distance, at $\beta=2$, for a three-stage extension of K-Means to alternatingly minimize the K-Means summary distance criterion over the three groups of variables: entity memberships, cluster centroids, and feature weights. Note that at β=2, the weights themselves have the meaning of feature rescaling factors, that is, they can be put as part of the data pre-processing independently of the clustering criterion. Frigui and Nasraoui [18] introduced cluster-specific feature weights. However, they do not go beyond β=2 either and, moreover, change the criterion by introducing additive "regularization" terms. Huang et al. [6, 7] moved on with a further extension of K-Means criterion by admitting an arbitrary exponent β but yet utilized the squared Euclidean distance as the dissimilarity measure. They also changed the rule for expressing the extent of success: it is not the summary distance to centroids anymore, but rather the recovery of a "natural" pre-specified set of clustering labels. The change might be well justified by the impossibility of comparing the values of the summary distance criterion at different values β. This extension works well in practice; Huang et al. [5 - 7] show that most appropriate β values are data specific but they fall short of indicating how the value of β can be chosen in situations when class labeling is not known in advance. Another issue with this approach is that under the exponent $\beta\neq 2$ the weights are not the feature scale factors anymore. It is this shortcoming that is going to be addressed here by extending the Euclidean squared distance to Minkowski metric of the same power β. Also, a semi-supervised approach for finding an appropriate value of the exponent is to be tried (see later in section 4.3).

(3) Minkowski metric and other distances

Minkowski p-metric between M-dimensional points $x=(x_v)$ and $y=(y_v)$ is defined by equation

$$d_p(x,y)=\left(\sum_{v=1}^{M}\left|x_v-y_v\right|^p\right)^{1/p}. \qquad (1)$$

In most applications, only values $p=2$ (Euclidean metric), p=1 (Manhattan, or City-block, metric) and $p\to\infty$ (Chebyshev, or Maximum, metric) have been considered. However, a few papers recently have been published on the usage of other values of p. Doherty et al. [19] note that data normalization differently improves cluster recovery with K-Means at different p values. Rudin [20] and Francois et al. [21] point to the differences in effects of data concentration at different p and possibilities of utilizing them for effectively tackling data analysis problems; the latter paper cites a number of useful mathematical properties of Minkowski metric. Kivinen et al. [22] undertake a comprehensive modification of the least-squares approach in prediction by using Minkowski p-metric. They point to a general fact that the metric allows to identify and, in fact, ignore irrelevant features, which goes in line with the main subject of this paper.

A wide class of different metrics have been introduced by Banerjee et al. [23]. Even wider classes of metrics may emerge in the context of kernel versions of K-Means, since the wealth of applicable kernels is virtually infinite (for a review, see [24]). However, comparative analysis of different metrics does not attract much attention as yet except in some specific areas such as

web-page clustering (see, for example, Strehl et al. [25]).

2.2. The number of clusters and initialization of K-Means

In spite of the well known and well documented fact that the local minima of K-Means criterion are not necessarily deep enough, there is not much known of the proper ways of initialization of the K-Means process, except probably for the information that multiple runs from seeds located at random data entities tend to converge to deeper minima than those starting from random seeds located anywhere in the data space (see Steinley and Brusco [26] for a review and discussion). A dozen of other initializing options considered by Steinley and Brusco [26] did not do terribly well in the experiments, except for the case of using Ward agglomerative algorithm to produce K clusters to initialize K-Mean. This is no wonder since Ward algorithm optimizes the same criterion; it just uses a different strategy.

The problem of choosing the right number of clusters at K-Means attracted much more attention (see, for example, [27] – [32], [10]). A number of these approaches, including those in [27]-[29], have been tested on synthetic data sets comprising a number of Gaussian clusters, possibly well elongated and intermixed, by Chiang and Mirkin [10]. In these experiments, a "rule of thumb" by Hartigan [32] based on the relative increment of the summary distance K-Means criterion proved superior in most cases. Another winner, especially in terms of the initial centroids, was a method from [3] utilizing an attractive idea of putting the initial centroids far enough from each other as anomalous patterns. This idea has been differently formalized in the literature including: (a) the MaxMin procedure [3, 26], (b) the algorithm Build preceding a popular version of K-Means, PAM method in which centroids are presented by central objects rather than by means [33], and (c) Anomalous clustering [3, 10], in which anomalous clusters are built as those most distant from grand mean and extracted one by one. The MaxMin builds a series of faraway initial seeds rather straightforwardly, whereas the other two utilize a more balanced approach by taking in only those distant objects that have a dense neighborhood. The MaxMin and Build have no natural stopping criterion and, thus, require the number of clusters K pre-specified. In contrast, the Anomalous clustering does have a natural stopping criterion – when no unclustered objects remain, and, therefore, can be used for finding "the right number of clusters" as well. The MaxMin and a version of Anomalous clusters have been included in the experiments by Steinley and Brusco [26] with rather mediocre results. It should be mentioned in this regard that the successful procedure for Anomalous clustering in [10] differs from that utilized in [26]. In Chiang and Mirkin [10], all anomalous clusters are found first, after which only those with the largest numbers of entities are taken to initialize the K-Means. The number K is defined in [10] by a threshold value such that the Anomalous clusters containing less entities than the threshold value are dissolved. When K is pre-specified, only the largest K Anomalous clusters are taken to initialize K-Means. This concurs with the idea of centroids to have dense neighborhoods. In contrast, the version utilized by [26] takes the first K extracted anomalous clusters to initialize K-Means, even if some of them have just one or two entities. The initialization with K largest Anomalous clusters [10] will be examined with Minkowski's distances in this paper as well.

3. *K*-Means algorithm and its weighted versions

This section describes the generic K-Means method and its extensions with respect to the weighting of the variables and initialization, both known, in the first subsection, and the newly

proposed ones, in the second subsection.

3.1. Versions of K-Means

In this subsection, three versions of K-Means are described, to be used for deriving the newly proposed versions of K-Means.

3.1.1 Generic K-Means

The *K*-Means clustering method in its generic version, the so-called batch mode, applies to a dataset involving a set of *N* entities, *I*, set of *M* features, *V*, and a quantitative entity-to-feature matrix $Y=(y_{iv})$, where y_{iv} is the value of feature $v \in V$ at entity $i \in I$. The method produces a partition $S=\{S_1, S_2, ..., S_K\}$ of *I* in *K* non-empty non-overlapping subsets S_k, referred to as clusters, each represented by a centroid $c_k=(c_{kv})$, an *M*-dimensional vector in the feature space $(k=1,2,...K)$. The criterion, alternatingly minimized by the method, is the sum of within-cluster distances to centroids:

$$W(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(i, c_k) \qquad (2)$$

where $C=\{c_1, c_2, ..., c_K\}$ is the set of all centroids and $d(i,c_k)$ is a dissimilarity measure between *Y*'s *i*-th row and centroid c_k, usually taken to be the squared Euclidean distance, so that the criterion (2) can be rewritten as

$$W(S,C) = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} s_{ik} (y_{iv} - c_{kv})^2 \qquad (3)$$

where s_{ik} is a binary cluster membership variable such that $s_{ik}=1$ if $i \in S_k$ and $s_{ik}=0$, otherwise. Starting from *K* initial *M*-dimensional cluster centroids c_k, the K-Means algorithm updates clusters S_k according to the Minimum distance rule: For each entity *i* in the data table, its distances to all centroids are calculated and the entity is assigned to its nearest centroid. Then centroids $\boldsymbol{c_k}$ are updated, given clusters S_k, according to the distance *d* in criterion (2), $k=1, 2, ..., K$. With the criterion in the form of equation (3), c_k is calculated as the vector of within-cluster averages that minimize (2) given the cluster membership. This process is reiterated until the clusters stabilize.

3.1.2 Choice of K and initial setting: iK-Means

The "intelligent" version of K-Means, iK-Means, described in [3,10] utilizes the so-called anomalous clusters that are found before running K-Means itself. Anomalous clusters are extracted one-by-one till no unclustered objects remain, after which centroids of the largest anomalous clusters are used to initialize K-Means [3, 10]. Each of the anomalous clusters is built by taking, as its initial centroid, the entity that is farthest away from a pre-specified "reference point". Then the anomalous cluster is filled in by the entities that are nearer to it than to the reference point, which is iteratively updated by updating the center of the anomalous cluster in the manner of K-Means itself, until the cluster stops changing. The resulting anomalous cluster is removed from the data set, and the procedure is repeated with the reference point unmoved. When all entities have been clustered in this way, the centroids of non-singleton anomalous clusters are used to both set K and initialize K-Means. When the number K is pre-specified, the procedure is modified accordingly: centroids of only K largest anomalous clusters are taken to

initialize K-Means. In the absence of other information, the reference point is set as the central point of the data set, that minimizing the summary distance to all data entities, which is "grand mean", the gravity center, when the dissimilarity d is Euclidean squared distance.

In a series of experiments with generated Gaussian clusters of varying degrees of within- and between- cluster spreads and thus overlaps, iK-Means has outperformed, in most cases, many other K selection algorithms in terms of both cluster recovery and centroid recovery [10].

3.1.3 Feature Weighting

Huang et al. [5, 6, 7] modify criterion (3) to include unknown weights w_v of the variables v, $v=1, \ldots M$:

$$W(S,C,w)=\sum_{k=1}^{K}\sum_{i\in I}\sum_{v=1}^{M} s_{ik} w_v^{\beta} (y_{iv} - c_{kv})^2 \qquad (4)$$

where the feature weights w_v are supposed to be non-negative and sum to unity. The exponent β is a user-defined parameter that expresses the rate of effect of the weight on its contribution to the distance.

To minimize criterion (4) over unknown clusters, centroids and feature weights, Huang et al. [5, 6, 7] propose a process, referred to as WK-Means, of alternating minimization in iterations similar to those in K-Means but involving three steps according to the number of groups of variables in criterion (4): clusters S_k, centroids c_k and weights w_v. These steps are:

(i) given centroids and weights, update clusters by using the minimum distance rule where distance is modified to include the weights: $d(y_i,c_k)=\sum_{v\in V} w_v^{\beta} (y_{iv} - c_{kv})^2$;

(ii) given clusters and weights, update centroids conventionally as clusters' gravity centers, that is, means;

(iii) given clusters and centroids, update weights according to the formula followed from the first order optimality condition for the problem of minimization of (4) constrained by the condition that the sum of weights is 1:

$$w_v = \frac{1}{\sum_{u\in V} [D_v / D_u]^{\frac{1}{\beta-1}}} \qquad (5)$$

where $D_v = \sum_{k=1}^{K}\sum_{i\in I} (y_{iv} - c_{kv})^2 s_{ik}$, the sum of within-cluster variances of feature v weighted by clusters' cardinalities.

Formula (5) much resembles a similar formula for the entity membership values in the fuzzy c-means algorithm [8]. The reason is the structure of criterion (4), similar to that of the c-means criterion. This makes the minimizer of (4) with respect to the linear constraint $\sum_{v\in V} w_v=1$ to satisfy two additional properties. First, the solution is expressed in a closed form (5), and, second, all the optimal weights are positive. Formula (5) is not applicable when $D_u = 0$ for some $u \in V$. To alleviate the issue, Huang et al. [5, 6, 7] advise adding a positive value, the average feature variance, to each D_u in formula (5). The other case at which formula (5) is not applicable is of $\beta=1$, as this would generate a division by zero in the exponent. In this case the minimum of (4) is reached at $w_{v*}=1$ for the feature v* with the smallest sum of the within-cluster distances

and all other feature weights set to 0 [7].

The WK-Means converges in a finite number of steps because at each step the criterion decreases, whereas the number of different partitions is finite.

Huang et al. [7] further extend criterion (4), and WK-Means, to the case of "subspace clustering", at which the variable weights are cluster-specific. The extension is rather straightforward – all the steps and formulas remain the same, except that the weights w_v and distances D_v in (4) and (5) become cluster-specific w_{kv} and D_{kv}, respectively. The latter is provided by dropping the sum over k from the definition of D_v in (5):

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv} / D_{ku}]^{\frac{1}{\beta - 1}}} \qquad (5')$$

where $D_{kv} = \sum_{i \in I} (y_{iv} - c_{kv})^2 s_{ik}$, which is the within-cluster variance of feature v weighted by the cluster's cardinality.

3.2. Further extensions of Weighted K-Means clustering

In each of the three following subsections, a new version of weighted K-Means is proposed. The section ends with a brief discussion of computational issues related to the proposed algorithms.

3.2.1. Weighted iK-Means

The anomalous clusters step can be rather straightforwardly implemented for the weighted distance criterion (4) to set K and initialize centroids before application of the WK-Means method. This will be referred to as iWK-Means:

0. Normalize the data.
1. Initialize the weights to be equal to each other in all the features.
2. The non clustered entity farthest away from 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, in terms of the weighted distance, to the tentative centroid than to the origin 0;
    b. Update the centroid to the mean of S;
    c. Update the weights for this centroid by using equation (5) or (5′);
    d. If the new centroid differs from the previous one, go to step a; otherwise stop and remove S from the dataset.
3. Repeat step 2 until all the entities are clustered.
4. Select the centroids of the K largest clusters.
5. Run WK-Means starting from the found centroids and weights.

3.2.2. Minkowski metric Weighted K-Means

The feature weights of WK-Means method can be associated with feature scaling factors if the distance in WK-Means criterion (4) is extended from the squared Euclidean to Minkowski β-metric in criterion (6):

$$W_\beta(S,C,w) = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} s_{ik} w_v^\beta \mid y_{iv} - c_{kv} \mid^\beta = \sum_{k=1}^{K} \sum_{i \in I} \sum_{v=1}^{M} s_{ik} \mid w_v y_{iv} - w_v c_{kv} \mid^\beta = \sum_{k=1}^{K} \sum_{i \in S_k} d^\beta(y_i', c_k') \qquad (6)$$

Criterion (6) differs from criterion (4) only in the distance exponent, β rather than 2, so that the right-hand expression refers to β power of Minkowski β-metric (1) between the rescaled entity points $y_i'=(w_v y_{iv})$ and centroids $c_k'=(w_v c_{kv})$ in the feature space. Yet criterion (6) returns the weighted version to the original K-Means formulation, the right-hand expression in (6), by identifying the weights with the feature scaling coefficients.

Minimization of Minkowski metric K-Means score function (6) follows that of the weighted function (4), with some changes due to the distance exponentβ. Specifically, the alternating minimization in iterations over the three groups of variables – clusters, centroids and weights – is similar to that in WK-Means. Here is a formulation of the algorithm, further referred to as MWK-Means, for alternating minimization of criterion (6) at a given β:

(i) given centroids c_k and weights w_v, update the cluster assignment of entities by using the Minimum distance rule with distance defined as β power of Minkowski β-metric in (6),
$d^{\beta}(y_i, c_k) = \sum_{v=1}^{M} w_v^{\beta} | y_{iv} - c_{kv} |^{\beta}$;
(ii) given clusters S_k and weights w_v, update centroid $c_k = (c_{kv})$ of each cluster S_k as its Minkowski center so that, at each v, c_{kv} is defined by a value c minimizing an item in Minkowski's distance β power,

$$d(c) = \sum_{i \in S_k} | y_{iv}' - c |^{\beta}; \qquad (7)$$

where $y_{iv}'=(w_v y_{iv})$, according to equation $c_{kv}=c /w_v$ as follows from the notation introduced for equation (6). (Indeed, since criterion (6) is the sum of independent items of the form of (7) so that an optimal set of centroids C consists of values c_{kv}, each minimizing (7) for the corresponding $v \in V$ and $k=1,..., K$.)
(iii) given clusters S_k and centroids c_k, update weights according to the formula followed from the first order optimality condition for the problem of minimization of (6) constrained by the condition that the sum of weights is 1:

$$w_v = \frac{1}{\sum_{u \in V} [D_{v\beta} / D_{u\beta}]^{\frac{1}{\beta - 1}}} \qquad (8)$$

where $D_{v\beta} = \sum_{k=1}^{K} \sum_{i \in I} | y_{iv} - c_{kv} |^{\beta} s_{ik}$.
This algorithm indeed is an algorithm of alternating minimization for criterion (6) over the three groups of the variables: the feature weights, the centroids, and the clusters. The starting point is a random selection of centroids and equal weights of the variables, after which the iterations of the alternating minimization are run, each as a sequence of steps (i), (ii), and (iii) as described above.

To prove that formula (8) gives the optimal weights, given the centroids and clusters, let us rewrite the Minkowski criterion (6) as $W_{\beta}(S,C,w) = \sum_{v=1}^{M} w_v^{\beta} D_{v\beta}$. To minimize this over w_v 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 derivative 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 $1 = \sum_{v} (\lambda / \beta D_{v\beta})^{\frac{1}{\beta-1}}$, so that $(\lambda / \beta)^{\frac{1}{\beta-1}} = 1 \Big/ \sum_{v} (1 / D_{v\beta})^{\frac{1}{\beta-1}}$. This leads to equation (8) indeed.
Note that (8) guarantees that the weights cannot be negative, which has not been required in the problem solved by the equation, and, in fact, is a bonus implied by the additive structure of the criterion (6). Of course, equation (8) is not applicable when $D_{u\beta} = 0$ for some feature $u \in V$.

We follow a further extension of WK-Means algorithm proposed by Huang et al. [5-7] who were using cluster-specific weights (5′) rather than those cluster independent ones, as described in section 3.1.3. These are derived similarly in the Minkowski metric context, so that cluster-specific weights are computed according to formula

$$w_{kv} = \frac{1}{\sum_{u \in V} [D_{kv\beta} / D_{ku\beta}]^{\frac{1}{\beta-1}}} \qquad (8')$$

where the summary distance $D_{kv\beta} = \sum_{i \in I} | y_{iv} - c_{kv} |^{\beta} s_{ik}$ is cluster-specific.
Another issue of the MWK-Means algorithm is finding Minkowski's centers whose components are minimizers of the summary Minkowski distances (7). The problem is to find a real value c minimizing the summary distance (7) for a given set of real values representing a feature within a cluster. A computationally feasible procedure for computing Minkowski's center of a set of values y_i, $i=1,2,..., n$, at any β can be defined with a nature-inspired evolutionary algorithm. However, in the case of $\beta \geq 1$, the only one that is explored here, a better option is available. Indeed, the summary distance $d(c)$ in (7) is a convex function of c at $\beta \geq 1$. Then a steepest descent procedure can be applied to find the global minimizer. As it is well known, at $\beta=1$, the median minimizes the distance $d(c)$ in (7), so that further on only $\beta > 1$ are considered. Assume that the y-values are sorted in the ascending order so that $y_1 \leq y_2 \leq ... \leq y_n$. Let us first prove that the optimal c must be between the minimum, y_1, and the maximum, y_n, of the range. Indeed, if, on the contrary, the minimum is reached outside of the interval, say at $c > y_n$, then $d(y_n) < d(c)$ because $|y_i - y_n| < |y_i - c|$ for all $i=1,2, ..., n$; and the same holds for the β-th powers of those. This contradiction proves the statement. To prove the convexity, consider any c in the interval between y_1 and y_n. Distance function (7) then can be rewritten as

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

where I^+ is set of those indices $i=1,2,..., n$ for which $c > y_i$, and I^- is set of such i's that $c \leq y_i$.

Then the first derivative of $d(c)$ can be expressed as $d'(c) = \beta(\sum_{i \in I^+}(c - y_i)^{\beta-1} - \sum_{i \in I^-}(y_i - c)^{\beta-1})$, and the second derivative, as $d''(c) = \beta(\beta-1)(\sum_{i \in I^+}(c - y_i)^{\beta-2} + \sum_{i \in I^-}(y_i - c)^{\beta-2})$. The latter expression is positive for each c value, provided that $\beta>1$, which proves that $d(c)$ is convex indeed.

The convexity leads to one more useful property: assume that $d(y_{i*})$ is the minimum among all n $d(y_i)$ values ($i=1, 2, ..., n$) and denote by $y_{i'}$ the maximum of y_i-values such that: (a) $y_i<y_{i*}$ and (b) $d(y_i)>d(y_{i*})$. Similarly, denote by $y_{i''}$ the minimum y_i-value such that: (a) $y_i>y_{i*}$ and (b) $d(y_i)>d(y_{i*})$. Then the minimum of $d(c)$ lies within the interval $(y_{i'}, y_{i''})$.

These properties justify the following steepest descent algorithm for finding Minkowski's center of a set $\{y_i\}$ of values $y_1 \le y_2 \le ... \le y_n$ at $\beta>1$:

Minkowski center algorithm

1. Initialize with $c_0=y_{i*}$, the minimizer of $d(c)$ on the set $\{y_i\}$ and a positive learning rate λ that can be taken, say, as 10% of the range $y_n - y_1$.
2. 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 a bit, say, by 10%, and repeat the step.
3. Test whether c_1 and c_0 coincide up to a pre-specified precision threshold. If yes, halt the process and output c_1 as the optimal value of c. If not, move on.
4. Test whether $d(c_1) \le d(c_0)$. If yes, set $c_0=c_1$ and $d(c_0)=d(c_1)$, and go to step 2. If not, decrease λ a bit, say by 10%, and go to step 2 without changing c_0.

In the computations, this method appears to converge much faster than a nature-inspired evolutionary method (not described here).

What is said above suggests an iteration in the MWK-Means algorithm with cluster-specific feature weights outlined as follows:

(i) given centroids and weights, update clusters by using the minimum distance rule where distance is: $d(y_i,c_k)=\sum_{v \in V} w_{kv}^{\beta}(y_{iv} - c_{kv})^2$;

(ii) given clusters and weights, update centroids component-wise by using Minkowski center algorithm;

(iii) given clusters and centroids, update weights according to formula (8´).

3.2.3. Minkowski metric Weighted iK-Means

The MWK-Means algorithm initialized with centroids of anomalous clusters found by using Minkowski metric and the found feature weights will be referred to as iMWK-Means. It should be noted that on the second step of the anomalous pattern algorithm the reference point is defined now as the Minkowski's center of the entity set. In step 2b, the mean of S must be replaced by the Minkowski center of S, and in step 2c, equations (8) and (8´) are used instead of (5) and (5´), respectively.

In relation to $D_{v\beta}=0$ in (8) and $D_{kv\beta}=0$ in (8´), in our experiments, it appears that adding a very small constant to the denominator instead of the average feature variance, advised by Huang et al. [5 – 7], tends to increase the accuracy. This applies only at the initialization stage of the algorithm; MWK-Means remains then unchanged.

3.2.4. Computation time issues

In spite of the popularity of K-Means, its computational properties are not well known. Obviously, the computation time at each iteration is proportional to the product of the number of entities and the dimension of the variable space. The number of iterations in practice is rather small; yet a recent paper [34] suggests that, in a worst-case scenario, this can be quite high. The weighting should not much change that - using the formulas (5) and (5′) may just somewhat increase the scaling coefficient at the running time and, probably, increase the number of iterations. The proposed modifications may lead to more complex computations yet. The Anomalous cluster procedure at i-versions should not much increase the total time: the added computation of centroids should lead to a smaller number of iterations because the choice of centroids is dictated by a version of K-Means criterion, so that, on the balance, this should not be of an issue. However, the computation of Minkowski's centers can slow down the running time drastically because it involves an iterative steepest descent process the number of iterations at which can be significant.

4. Experiments with Weighted K-Means algorithms

Experiments on the proposed methods are conducted in the following three directions:

(1) Testing the proposed Minkowski metric algorithms for cluster recovery, especially in the presence of noise features, in comparison with the original method by Huang et al. [5-7];
(2) Exploring the behavior and accuracy of the proposed Minkowski metric algorithms with respect to increasing the numbers of features;
(3) Exploring the possibility of estimating the value of the exponent β in a semi-supervised manner.

Relative running times of the algorithms are of interest as well.

The section is divided in subsections accordingly.

4.1. Testing Minkowski metric Weighted K-Means algorithms

The goal of this section is to experimentally explore the performance of the Minkowski metric Weighted K-Means algorithms at best possible values of the Minkowski exponent, in relation to the best performances of other K-Means algorithms under consideration.

Algorithms under comparison and datasets utilized are described below. The scoring criterion is the accuracy in recovery of pre-assigned cluster labels.

4.1.1. Algorithms under comparison

Basically, the proposed MWK-Means algorithm is compared with WK-Means and generic K-Means, so that two versions of each are taken: one based on a hundred runs starting from a random initialization, and the other, starting from an anomalous cluster initialization. This leads to the following six versions of K-Means (at pre-specified K):

1. K-Means: Results of a hundred runs of generic K-Means at random initializations;
2. WK-Means: Results of a hundred runs of the Weighted K-Means algorithm by Huang et al. at the best β values, yet with the same β at all of the hundred runs;
3. MWK-Means: Results of a hundred runs of the Minkowski metric Weighted K-Means at the best β values, yet with the same β at all of the hundred runs;
4. iK-Means: Generic K-Means initialized at the centroids of K largest anomalous clusters;
5. iWK-Means: Weighted K-Means algorithm initialized at the centroids of K largest anomalous clusters, at the best β value;

6. iMWK-Means: Minkowski metric Weighted K-Means algorithm initialized at the centroids of K largest anomalous clusters, at the best β value.

In all the experiments the range of β values considered is from 1 to 5, with the precision of up to one decimal digit.

In the weighted versions all the feature weights are taken to be cluster-specific.

Before running an algorithm, the dataset is pre-processed so that every feature is standardized by subtracting its average from the data entries and dividing the result by half the feature's range, the difference between the maximum and minimum divided by 2.

4.1.2. Datasets for the experiments

With respect to data sets in the experiments, both real datasets and synthetic ones, list of sets analyzed by Huang et al. [5 – 7] is taken and somewhat extended. Specifically, six real datasets come from the UCI Machine Learning Repository [11]: two of them, the Australian credit card and Heart disease datasets have been analyzed by Huang et al. [7], the other four are popular datasets, Iris, Wine, Hepatitis, and Pima Indian Diabetes; all with pre-labeled clusters. Some of these, like Pima Indian Diabetes, have rather complex class structure so that none of the existing classification algorithms have achieved anything better than 70-80% accuracy. It should be noted, that all the selected algorithms are for clustering, not classification. In the setting of the experiments, only one parameter, the feature weight exponent β, is subject to adjustment to the pre-specified class labels as it has been in the work by Huang et al. [5 – 7]. This drastically differs from the adjustment of classification algorithms in which the number of parameters is not one or two but often exceeds that of the number of features. That means that, with adjusting just one parameter, there will likely not be much overfitting, which may and do happen when a classification algorithm is run. At this stage, those values of β at which this or that algorithm performs best are used, as it has been in the work of Huang et al. [5 – 7]; the issue of learning β from the data will be dealt with later, in section 4.3.

All synthetic data sets in this section have been generated as a 500 strong six-dimensional GM (Gaussian Model) data set, each comprising five Gaussian clusters with the mixture coefficients equal to 0.2 each, by using the Netlab software [35]. Each of the clusters is spherical of the variance 0.1; its center's components are independently generated from a Gaussian distribution N(0,1) of zero mean and unit variance.

The tests are run on both the datasets as they are and on versions of the data with added noise features. A noise feature is generated, after data standardization, as a uniformly random distribution in the interval of the length 2, [-1,+1], similar to those of the features innate to the data set.

4.1.3. Result scoring function

The evaluation of the algorithm's results follows Huang et al. [5 –7] by using the accuracy, the proportion of points correctly clustered by an algorithm under consideration. Specifically, each of the K clusters produced by the algorithm is mapped to that of the pre-labeled K clusters, that makes the largest overlap; the set of points in that largest overlap is stored for each cluster, these overlap sets are merged together, and the size of the merged set is computed proportional to the size of the whole data [5 - 7]. This proportion is taken as the accuracy. The values of the K-Means criterion itself are not used because they are not comparable at different β's.

4.1.4 Results on data with no added noise

In this subsection, the experiment goes over real and synthetic data sets as they are, with no noise features added to them.

4.1.4.1 Analysis of the Iris dataset

The Iris dataset of 150 flower specimens characterized by 4 features (Sepal Length, Sepal Width, Petal Length, Petal Width) consists of 3 clusters representing three Iris species, 50 specimens in each.

Results of the six algorithms under consideration on the Iris data set are in Table 1. To express all the different criteria within a uniform framework, we consider them as special cases of the Weighted K-Means criterion at which the exponent values at the weights and the point distances may differ – these are put here and further on as "Exponent β at weight" and "Exponent β at distance", respectively. Following Huang et al. [5] the values of β exponent adjusted at the best accuracy of cluster recovery are reported. Also, the maximum accuracy (over random intilializations, for methods that use them) as well as the average accuracy and its standard deviation is presented. The average accuracy gives an overall performance measure that indicates the expected performance of the algorithm under consideration in the situations at which classes are not pre-labeled – these are of utmost interest to a data miner. This model of reporting is carried through all the further results described in the paper.

[TABLE 1 HERE]

One can see that the best results, 96.7% accuracy, only 5 misclassified objects –have been achieved by the Weighted iK-Means in both versions – at the Euclidean and Minkowski distance metric with respectively β = 1.1 and 1.2. The same accuracy was achieved once in a hundred of random runs of Weighted K-Means.

This favorably compares with the accuracy achieved at the Iris using different clustering or classification algorithms as reported in the recent literature (see Table 2), although one should not forget that classification algorithms are tested differently, by using test sets that have no overlap with the training sets.

[TABLE 2 HERE]

To see what can cause a successful clustering, let us take a look at the feature weights in Table 3 at which rows correspond to clusters and columns to features.

[TABLE 3 HERE]

As one can see from Table 3, features 3 and 4 (Petal Length and Petal Width) have considerably higher weights in all clusters. This is well aligned with the literature which tends to show these features – or their product, as being the informative ones. Yet here one can observe that the relative weights of these differ from cluster to cluster.

4.1.4.2 Analysis of the Wine dataset

Wine dataset comprises results of chemical analyses of various wine probes over 13 features; the 178 specimen of wine are divided in 3 classes according to their production region.

Table 4 presents the results of the algorithms on the Wine dataset. In contrast to the Iris dataset case, the introduction of feature weights here does not lead to improving the accuracy – the

generic K-Means here leads to very high results.
[TABLE 4 HERE]

Table 5 presents a sample of best results by other algorithms. One can see that they can achieve as much as the best run of generic K-Means, 96.6%, with iWMK-Means reaching almost as much, 94.9%. Another criterion is taken by Tsai and Chiu [17]: their experiments on Iris and Wine datasets lead to rather low levels of the adjusted Rand coefficient, about 0.77 and 0.40, respectively [17, p. 4667] – these correspond to the accuracy levels way below those reported above in Tables 1-2 and 4-5.

[TABLE 5 HERE]

4.1.4.3 Analysis of the Hepatitis dataset
This is a dataset of 155×19 size. Many of the features are yes-no binary, which are coded here with 1-0 values, so that all features are considered to be quantitative as described in [3]. There are two pre-labeled classes, "live" and "die". The accuracy levels achieved on the Hepatitis dataset are recorded in Table 6. Here, iMWK-Means outperformed all the other K-Means versions with the accuracy 84.5%.

[TABLE 6 HERE]
This result rather favorably compares with results recently published in [43, 44] where the training extends much further, to produce classifiers rather than clusterers (see Table 7), though, as already mentioned, the classifiers are tested more thoroughly, on the parts of the datasets that have not been used for training the algorithms.

[TABLE 7 HERE]

4.1.4.4 Analysis of the Pima Indians Diabetes dataset

[TABLE 8 HERE]

This dataset is of 768×8 size and involves two classes. The results in Table 8 show that Minkowski's metric based algorithms outperform the others, including Weighted K-Means, both in terms of maximum and average values achieved. The accuracy achieved is not high, though, and can be beaten with techniques involving further readjustment of clusters to minimize the number of misclassified entities [46] (see Table 9 in which the best results found after 50 random initializations in [46] are reported).

[TABLE 9 HERE]

4.1.4.5 Analysis of the Australian credit card and Heart Disease datasets

Tables 10 and 11 present results of the comparison between the weighted versions of K-Means method and the results achieved by Huang et al [5] on the Australian credit card dataset and Heart disease dataset. In contrast to the other datasets under consideration, these combine both

categorical and quantitative features so that Huang et al. [5] apply to them a somewhat differing method combining K-Means and K-Prototype in which centroids are represented by averaged quantitative values and modal categorical values, respectively. To remain within the original K-Means framework, a different strategy is utilized here. According to this strategy [3], the data are quantified by representing each categorical feature by a quantitative binary variable, a dummy that assigns 1 to each entity which falls in the category and 0 if not. Each of these variables is then standardized in a quantitative way by subtracting its grand mean, that is, the category's proportion. That means that centroids are represented by the proportions rather than just the modal values. Because of the different standardizations, the clustering results may differ too. Therefore, for the weighted versions of K-Means, Tables 10 and 11 present both the results reported by Huang et al [5] after working of their combined K-Means/Prototype algorithm, in the first row, and the results found with an in-house implementation of the WK-Means method, in the second row. Another issue is that the currently available version of Australian credit card dataset is used. It contains records of 690 instances in which all missing values have been substituted by the feature's average values [11], whereas Huang et al. [5] take only 666 of them, those with no missing values.

[TABLE 10 HERE]

As can be seen from Tables 10 and 11, given the specific data coding, iWMK-Means's results are superior to those of WK-Means, and competitive with the best results achieved by the WK-Means/Prototype method by Huang et al. [5].

[TABLE 11 HERE]

Modha and Spangler [16] apply their method to Australian credit card dataset by separating the subspace of quantitative features from the subspace of dummy variables related to the categorical features and assigning thus only two weights pertaining to the subspaces rather than to individual features. There is no wonder thus that the accuracy achieved in [16], about 83% at K=2, is somewhat inferior to those reported in Table 10 for WK-Means and iMWK-Means, 85% and 86%, respectively.

4.1.5 Experiments on synthetic data

Ten 500 strong six-dimensional GM (Gaussian Model) data sets, each comprising five Gaussian clusters with the mixture coefficients equal to 0.2 for each, have been generated by using Netlab software [34]. Each of the clusters is spherical of the variance 0.1; its center's components are independently generated from a Gaussian distribution N(0,1) of zero mean and unity variance. The averaged results of the experiment are presented in Table 12.

[TABLE 12 HERE]

All the within-cluster feature weights are more or less similar, which corresponds to the way the data are generated, with no preference assigned to any feature or cluster. This cluster structure gives no special advantages to weighted or anomalous clusters, which is manifested in the fact that on the level of the best results, as well as on average, all methods work similar. This is in stark contrast to the results obtained on the real world datasets from the UCI Machine Learning

Repository where intelligent versions of K-Means outperformed the conventional multiple run schemes, probably because the "anomalous" clusters model is more adequate to real data than to those generated as described above.

4.1.6. Results on data with noise features added

Huang et al. have shown that, unlike the generic K-Means, their weighted version of K-Means is rather robust against added noise features [5 – 7]. The aim of the experiments is to check whether Minkowski and intelligent versions of weighted K-Means do keep this advantage or they may be even better in this respect.

Specifically, a pre-specified number of uniformly distributed noise features are added to each of the datasets. For each dataset a pre-specified number of noise features is generated 10 times, so that there are 10 random copies processed with each of the clustering algorithms in the study. The reported figures are the averaged results of these computations.

The Iris and Wine datasets are supplemented with (a) as many noise features as there are in the data, or (b) half that number, which is 4 and 2 for Iris, and 13 and 7 for Wine, respectively. To illustrate how much the additional irrelevant features affect the data structures, Figures 1 and 2 illustrate the Wine and GM original datasets and their noisy versions projected to the plane of the two first principal components.

[FIGURE 1 HERE]
[CAPTION FOR FIGURE 1 HERE]

[FIGURE 2 HERE]
[CAPTION FOR FIGURE 2 HERE]

Curiously, both noisy versions of the Iris dataset lead to the same optimal parameter values at the weighted clustering, though results slightly differ (see Table 13). On the other hand, best results on the noisy versions of Wine set are achieved at differing parameters as shown in Table 14.

[TABLE 13 HERE]

The structure of feature weights on the noisy Iris data reproduces that at the original features giving near zero weights to all of the added noise features in each cluster.

[TABLE 14 HERE]

On the noisy Wine set (Table 14), the summary weight of the noise features on the anomalous clusters is less than 1% of the total, and it increases to about 10% of the total weight at the output of MWK-Means applied starting from those clusters' centroids, according to the iMWK-Means approach.

For each of the GM datasets a noisy version was created by adding two noise features (see Figure 2 demonstrating the structures of the 6-dimensional GM sets and their 8-dimensional

versions with noise features on the plane of the two first principal components: the clusters are hardly distinguishable on a noisy GM set).

[TABLE 15 HERE]

Table 15 presents results of applying the algorithms to the noisy versions of GM data sets that consistently show the following:

(i) The accuracy levels are much more modest here than when analyzing the data with no added noise (see Table 12). This is reflected in the fact that the optimized feature weights are all of the order of 0.13-0.16 for the six original features and of 0.05-0.06 for the two noise features. This again contrasts the almost zero weights of noise on real datasets.
(ii) On average the maximum accuracy of Minkowski metric K-Means is greater than that by WK-Means.
(iii) The original K-Means shows quite modest accuracy rates, reflecting the mentioned lack of structure on the synthetic data sets.
(iv) On average, the i-versions of the algorithms outperform the multiple runs of the corresponding versions of K-Means under investigation, with iMWK-Means achieving the best performance. This shows that in a situation of clustering unlabeled data the intelligent Minkowski metric Weighted K-Means version remains a viable option even when the data structure is rather vague.

Overall, these experiments show that Minkowski metric Weighed K-Means versions are quite competitive and frequently superior to Euclidean metric options on both the original datasets and their noisy versions.

4.2. The effects of increasing the feature space sizes

To see how the algorithms presented work at larger feature spaces, three further series of ever larger datasets have been generated, with cluster structures defined by NetLab software [35], as described in section 4.1.2 above, plus a number of noise features added afterwards:

(1) GM 5-cluster structure 500×15 datasets with 10 noise features added, amounting to 500×25 datasets;
(2) GM 12-cluster structure 500×25 datasets with 25 noise features added, amounting to 500×50 datasets;
(3) GM 12-cluster structure 1000×50 datasets with 25 noise features added, amounting to 1000×75 datasets.

The results for each of the series, averaged over ten datasets generated for each of them, are presented in respective Tables 16, 17, and 18.

[TABLE 16 HERE]

Table 16 presents a pattern similar to that of the Table 15, with Minkowski metric versions outperforming the others, and i-versions outperforming on average those based on multiple runs. The generic K-Means suffers most and shows rather poor performances.

[TABLE 17 HERE]

Table 17 shows a rather different pattern emerging. K-Means shows even a poorer performance, probably because the proportion of noise features here, one half, is the highest. Minkowski metric versions still outperform their Euclidean metric based counterparts, though, with rather mediocre results. However, the balance shifts towards the Weighted versions based on multiple runs: they outperform their corresponding i-versions.

[TABLE 18 HERE]
The change of the pattern is complete at the largest generated size series (3) presented in Table 18. Minkowski metric Weighted K-Means versions still outperform their Euclidean metric based counterparts. However, it is the generic K-Means that dominates here overall, reaching on some of the datasets 100% cluster recovery. In contrast, i-versions show relatively poor performances. All of these probably can be explained by the effects of the higher dimension sizes [47]: with all the distances relatively small and similar to each other, the "anomalous" patterns are no longer distinguishable, the individual variables are more or less of the same weight, and the fact that the irrelevant features prevail, making a majority of two thirds, gives the generic K-Means the ability of good cluster recovery.

4.3. Semi-supervised learning the Minkowski metric exponent value
To see whether the Minkowski metric β value can be learned from the data in a semi-supervised manner, a series of experiments involving most of the datasets utilized in the previous computations have been conducted for iMWK-Means method versus WK-Means. Specifically, the value of β has been learnt from exposing the class labels on a 20% data sample only, yet the clustering was conducted over either this very sample (A option) or over the entire dataset (B option).
Given a dataset, each of the experiments comprises fifty runs of the following procedure:
1 – Randomly choose a 20% sample of the data plus labels;
2 – Run a series of clustering experiments at different β values using only the chosen 20% of the data (A option) or the whole dataset (B option);
3 – At each of the options, pick the experiment and β with the highest accuracy;
4 – At A option, use the chosen β value to cluster the whole of the dataset (including the initial 20%); at B option, just use the result of the corresponding experiment.

[TABLE 19 HERE]
[TABLE 20 HERE]

At A option, the results have been stable (over different 20% samples), yet rather disappointing, as the β values learnt have been rather far away from those found at the entire dataset. However, B option has provided a better match between the β values learnt in the semi-supervised and fully supervised settings, which can be clearly seen in Tables 19 and 20. These tables report the accuracy results for WK-Means and iMWK-Means, respectively. One can easily see that, at B option, iMWK-Means clearly outperforms WK-Means on both counts: (i) the match between the semi-supervised and supervised β values and (ii) the accuracy. The only exception to this is the case of synthetic sets of the largest dimension 1000×75 at which WK-Means is superior. This goes in line with the already observed poor performances of iMWK-Means supervised learning

of β in this case: just the very concept of "anomalous" pattern looses its ground in these conditions.

4.4. The running time of the algorithms

To illustrate the discussion of computational intensity of various versions of K-Means in section 3.2.4, a series of experiments have been conducted on the generated GM datasets with recording the CPU running time of the K-Means algorithm versions under comparison (see Table 21). All the computations have been executed on the Intel Core-2 cpu 2.4GHz, Ram 2Gb under Windows 7 operational system on the MatLab, version 2010 which can take advantage of a Core-2 CPU.

[TABLE 21 HERE]

Of course, the timing can be quite different on different computing systems; however the data in Table 21 seem rather indicative of the relative speeds of convergence of the algorithms under consideration. As expected, both feature weighting and anomalous cluster search do not much add to the running time of an algorithm as compared to the generic K-Means runs. Moreover, these can increase the speed of computation in situations of high proportions of noise features in the data as, for example, at the GM dataset of 1000 points and 25 features with supplementary added 25 random noise features. Evidently, this may happen because, on this type of data, the generic K-Means requires a large number of iterations to converge, whereas the weighted and i-based versions may take much less iterations. The weighted versions change the space structure by drastically reducing the significance of noise, whereas i-versions start from better initial centroids following the data structure.

In contrast, Minkowski's metric algorithms take a hundred or more times longer to converge – this is the toll taken by the process of finding the Minkowski centers in each cluster at each iteration. A run of the iMWK-Means or MWK-Means takes a hundred times greater to converge than a run of WK-Means. However, to find a better initialization, WK-Means needs to be run about a hundred times, which balances the score: in practical computations, iMWK-Means and WK-Means take about the same time to run.

5. Conclusion

Weighting of the variables is a way to advance into the problem of clustering of datasets at which the cluster structure is blurred by the presence of irrelevant, or noise, features, as the Weighted K-Means method clearly demonstrates [5-7]. In this paper, two modifications of this method are proposed and their competitiveness is experimentally demonstrated.

The main contribution of this paper is the extension of the exponent β from the weights in the original Weighted K-Means method to the distances, in the form of Minkowski metric criterion (6). This returns the K-Means criterion to its original format of summary distances between entities and their cluster centroids and, also, makes the weights to be the feature rescaling coefficients. The Minkowski metric criterion does the job: in the experiments, it consistently improves the accuracy of the Weighted K-Means both at the original and noisy datasets. The issue remaining to be addressed in this regard, as it is with the original Weighted K-Means, is of determining the right value of β exponent. Applying a semi-supervised setting by training β on labeled subsamples appears to be a promising direction. Another possibility would lie in trying to identify characteristics of the data structures that relate to specific values of β.

A related contribution of this paper is the usage of anomalous cluster centers to initialize both centroids and feature weights in the "intelligent" versions of the Weighted K-Means. This proved effective at modest to moderate data sizes. An improvement with regard to real world datasets is not unexpected because real data structures probably do have something to do with the anomalous cluster mechanism [3], unlike the generated data with their "rounded" clusters. Yet i-versions prove superior on average at synthetic data as well. Moving to the high-dimensions realm, though, changes the entire perspective on weighting as shown in section 4.2 – and this deserves further investigation. The issue of a relatively slow convergence of Minkowski metric algorithms should be investigated further as well. One potential direction can be related to developing a kind of scale to relate the exponent β value and the likely position of Minkowski center in a sorted series of the feature values – we know already, that at β=1 the center is the median, thus, it is in the middle of the series. Such a development would allow to change the steepest descent process by a less computationally intensive sorting.

One more direction for future work would be in extending this approach to objects of complex structure to combine clustering with semi or fully supervised learning for both feature selecting and weighting. The specifics of the complex structure of the objects would suggest a two-step approach here so that features are selected first and readjusted later. The Minkowski metric approach will be extended to the work that we have started in the analysis of signals [48] and unstructured texts [49].

Acknowledgments. The authors are indebted to the anonymous referees whose thorough comments made us work harder to widen the scope of the experiments conducted, and improved the paper overall. BM thanks the Decision Choice and Analysis Laboratory and the Program for Fundamental Studies at the National Research University Higher School of Economics, Moscow, for partial financial support of his work.

References

[1] A. Jain, R. Dubes, Algorithms for Clustering Data, Prentice-Hall, 1988.

[2] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools, Morgan and Kaufmann, 2005.

[3] B. Mirkin, Clustering for Data Mining: A Data Recovery Approach, Chapman and Hall/CRC Press, Boca Raton Fl., USA, 2005.

[4] R. M. Haralick, L. G. Shapiro, Image regimentation techniques, Computer Vision, Graphics and Image Processing 29 (1) (1985) 100-132.

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

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

[7] Z. Huang, M. K. Ng, H. Rong, Z. Li, Automated variable weighting in k-means type

clustering, IEEE Transactions on Pattern Analysis and Machine Learning 27 (5) (2005) 657-668.

[8] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[9] V. Makarenkov, P. Legendre, Optimal variable weighting for ultrametric and additive trees and K-Means partitioning, Journal of Classification 18 (2001) 245-271.

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

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

[12] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognition Letters 31 (8) (2010) 651-666.

[13] E. Mueller, S. Guennemann, I. Assent, T. Seidl, Evaluating clustering in subspace projections of high dimensional data, Proceedings of the VLDB Endowment 2 (1) (2009) 1270-1281.

[14] E. P. Xing, A. Y. Ng, M. I. Jordan, S. Russell, Distance metric learning with application to clustering with side-information, in: Proceedings of the Conference on Advances in Neural Information Processing Systems, Vancouver, Canada, December 9-11, 2003.

[15] M. Bilenko, S. Basu, R. J. Mooney, Integrating constraints and metric learning in semi-supervised clustering, in: Proceedings of the 21st International Conference on Machine Learning (ICML-2004), 2004, pp. 81-88.

[16] D.S. Modha, W.S. Spangler, Feature weighting in K-Means clustering, Machine Learning 52 (2003) 217-237.

[17] C.Y. Tsai, C.C. Chiu, Developing a feature weight adjustment mechanism for a K-Means clustering algorithm, Computational Statistics and Data Analysis 52 (2008) 4658-4672.

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

[19] K. A. J. Doherty, R. G. Adams, N. Davey, Non-Euclidean norms and data normalization, in: Proceedings of European Symposium on Artificial Neural Networks, 2004, pp. 181-186.

[20] B. Rudin, The P-Norm Push: A simple convex ranking algorithm that concentrates at the top of the list, Journal of Machine Learning Research 10 (2009) 2233-2271.

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

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

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

[24] M. Filippone, F. Camastra, F., S. Rovetta. A survey of kernel and spectral methods for clustering, Pattern Recognition 41 (2008) 176–190.

[25] A. Strehl, J. Ghosh, R. J. Mooney, Impact of similarity measures on web-page clustering, in: Proceedings of AAAI Workshop on AI for Web Search, Texas, USA, July 30 – August 1, 2000.

[26] D. Steinley, M.J. Brusco, Initializing K-means batch clustering: a critical evaluation of several techniques, Journal of Classification 24 (2007) 99-121.

[27] R. Tibshirani, G. Walther, T. Hastie, Estimating the number of clusters in a dataset via the Gap statistics, Journal of the Royal Statistical Society B. 63 (2001) 411-423.

[28] S. Monti, P. Tamayo, J. Mesirov, T. Golub, Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data, Machine Learning 52 (2003) 91-118.

[29] G.J. McLachlan, N. Khan, On a resampling approach for tests on the number of clusters with mixture model-based clustering of tissue samples, Journal of Multivariate Analysis (2004) 90-1005.

[30] R. Mojena, Hierarchical grouping methods and stopping rules: an evaluation, The Computer Journal 20 (4) (1977) 359-363.

[31] D. Pelleg, A. Moore, Xmeans: Extending kmeans with efficient estimation of the number of clusters, in: Proceeding of the 17th International Conference on Machine Learning, 2000, pp. 727-734.

[32] J. Hartigan, Clustering Algorithms, Wiley, NewYork, 1975.

[33] L. Kaufman, P.J. Rousseeuw, Finding Groups in Data: An Introduction to Cluster Analysis, Wiley-Interscience, 1990.

[34] D. Arthur, S. Vassilvitskii, Worst-case and smoothed analysis of the ICP Algorithm, with an application to the k-means method, SIAM Journal on Computing 39 (2009) 766-782.

[35] Netlab Neural Network software, http://www.ncrg.aston.ac.uk/netlab/index.php, accessed 3

April 2010.

[36] L. Zhong, Y. Jinsha, Z. Weihua, Fuzzy C-Mean Algorithm with Morphology Similarity Distance, in: Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery 3 (2009) 90-94.

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

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

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

[40] R. Koggalage, S. Halgamuge, Reducing the number of training samples for fast Support Vector Machine classification, in: Neural Information Processing 2 (3) (2004) 57-65.

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

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

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

[44] Z. Zainuddin, W. K. Lye, Improving RBF Networks Classification Performance by using K-Harmonic Means, World Academy of Science, Engineering and Technology 62 (2010) 983-986.

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

[46] N. M. Zeidat, C. F. Eick, K-medoid-style clustering algorithms for supervised summary generation, in: Proceedings of the International Conference on Machine Learning, 2004, pp. 932-938.

[47] C.C. Aggarwal, A. Hinneburg, D. Keim, On the surprising behavior of distance metrics in high dimensional space, in: Proceedings of the 8th International Conference on Database Theory, 2001, pp. 420-434.

[48] R. C. Amorim, B. Mirkin, J. Gan. A Method for Classifying Mental Tasks in the Space of EEG Transforms. Technical Report BBKS-10-01, Birkbeck University of London, London,

2010.

[49] R. C. Amorim, An adaptive spell checker based on PS3M: Improving the clusters of replacement words, in: M. Kurzynski, M. Wozniak (Eds.), Computer Recognition Systems 3, Springer Berlin/Heidelberg, 2009, pp. 519-526.

Table 1: Accuracy levels for different versions of K-means clustering at the Iris dataset; means, standard deviations, maxima and minima are over the hundred of random initializations when applicable.

Algorithm	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std dev	Max	Min
K	2	0	84.0	12.3	89.3	52.0
WK	2	1.8	87.1	13.8	96.0	50.7
MWK	1.2	1.2	93.3	8.3	96.7	59.3
iK	2	0	88.7			
iWK	2	1.1	96.7			
iMWK at	2.0	2.0	94.7			
different β	3.0	3.0	90.0			
	1.2	1.2	96.7			

Table 2: Comparing clustering results achieved by various algorithms on the Iris dataset

	Accuracy, %	Comments
FCM + MSD	93.3	Combined fuzzy c-means and

		multidimensional scaling [36]
SWFCM	91.3	A weighted version of fuzzy c-means [37]
TILDE	94.0	Top-down induction of logical decision trees [38]
CEFS	92.56 ±2.96	Clustering ensemble based method [39]
SS-NMF	92.7	Semi-supervised non-negative matrix factorization [40]
Fast SVM	94.7	Supervised fast Support Vector Machine technique separately classifying each of the classes [41]
iMWK-Means	96.7	At $\beta = 1.2$

Table 3: iMWK cluster-specific feature weights in the Iris dataset at $\beta=1.2$

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

Table 4. Accuracy levels achieved by the six versions of K-Means at the Wine dataset; means, standard deviations, maxima and minima are over the hundred of random initializations when applicable.

Algorithm	Exponent at		Accuracy, %			
	Distance	Weight	Mean	Std dev	Max	Min
K	2	0	95.3	0.4	96.6	94.9
WK	2	4.4	93.9	0.8	94.9	91.6
MWK	2.3	2.3	92.6	1.3	96.1	89.3
iK	2	0	94.9			
iWK	2	1.2	94.9			
iMWK at	1.2	1.2	94.9			
different β	2.0	2.0	92.1			
	3.0	3.0	93.8			

Table 5: Accuracy achieved by other clustering algorithms on the Wine dataset

	Accuracy, %	Comments
CEFS+PBIL	87.07±0.21	Ensemble based method + Incremental algorithm [39]
SGKK-Means	91.60±2.12	Soft geodesic kernel K-Means [42]
NK-Means	95.8	A neighborhood based initialization for K-means [43]
SWFCM	96.6%	A weighted version of Fuzzy c-means [37]

Table 6. Accuracy levels achieved by the six versions of K-Means at the Hepatitis dataset; means, standard deviations and maxima are over the hundred of random initializations when applicable.

Algorithm	Beta	Mean	Std	Max
K-Means		0.7151	0.0136	0.7226
WK-Means	1.0	0.7872	0.0013	0.8000
MWK-Means	1.0	0.7902	0.0084	0.8000
iK-Means		0.7226		
iWK-Means	1.0	0.7871		
iMWK-Means	2.3	0.8452		

Table 7. Accuracy levels achieved by recent cluster based classification techniques at the Hepatitis

dataset.

	Accuracy, %	Comments
RBFC +HKM	79.3 (0.14)	Radial Basis Functions classification network trained by using Harmony K-Means results [44]
PROAFTN with RVNS 85.8	85.8	Prototype based fuzzy techniques multi-criterion decision method involving several manually set parameters
MSDD algorithm	80.8	Multi-stream dependency detection (MSDD) algorithm [45]
iMWK-Means	84.5	Proposed Minkowski metric based version

Table 8. Accuracy levels achieved by the six versions of K-Means at the Pima Indian Diabetes dataset; means, standard deviations and maxima are over the hundred of random initializations when applicable.

Algorithm	Beta	Mean	Std	Max
K-Means		0.6667	0.0055	0.6680
WK-Means	4.5	0.6450	0.0298	0.6628
MWK-Means	3.9	0.6818	0.0285	0.7135
iK-Means		0.6680		
iWK-Means	1.8	0.6471		
iMWK-Means	4.9	0.6940		

Table 9. Accuracy levels achieved by recent cluster based algorithms at the Pima Indian Diabetes dataset.

	Accuracy, %	Comments
PAM	65.6	Partitioning around medoids PAM [33] used in [46]
SPAM	77.2	PAM based strategy to minimize the number of wrongly assigned objects [46]
SRIDHCR	79.5	Insertion/deletion hill climbing to minimize the number of wrongly assigned objects [46]
iMWK-Means	69.4	Proposed Minkowski metric based version

Table 10. Results of different weighted partitioning methods on the Australian credit card

dataset; means, standard deviations, maxima and minima are over the hundred of random initializations when applicable.

	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std	Max	Min
WK-Means combined with Prototype [5]	2.0	9	71.93		85	71*
WK-Means	2.0	4.9	71.82	13.71	86.52	45.22
iWK-Means	2.0	1.8	85.51		-	-
iMWK-Means	1.8	1.8	86.09	-	-	-

* Huang et al. [5] report of 81 runs out of 100 leading to the accuracy of 71% or less; thus, the figure of 71% is an upper estimate of the minimum accuracy, also used to compute the mean accuracy.

Table 11. Results of different weighted partitioning methods on the Heart disease dataset; means, standard deviations, maxima and minima are over the hundred of random initializations when applicable.

	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std	Max	Min
WK-Means/Prototype [5]	2.0	9.0	73.55		85	71*
WK-Means	2.0	4.2	78.50	6.22	82.59	53.33
iWK-Means	2.0	3.8	80.37		-	-
iMWK-Means	2.7	2.7	84.07		-	-

* Huang et al. [5] report of 63 runs out of 100 leading to the accuracy of 71% or less; thus, the figure of 71% is an upper estimate of the minimum accuracy, also used to compute the mean accuracy.

Table 12. Average accuracy results for K-Means versions under consideration on the ten GM datasets. The best values of exponent β for MWK and iMWK versions are given for one of the GM sets; means, standard deviations and maxima are over a hundred of random initializations when applicable.

	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std	Max	
K-Means	2	0	79.8	9.1	89.4	
WK-Means	2	4.2	78.2	10.6	91.8	
MWK-Means	2.5	2.5	77.5	10.7	91.2	

iK-Means	2	0	82.6	7.7	89.4	
iWK-Means	2	4.5	81.0	9.9	89.8	
iMWK-Means at different β	2.4	2.4	81.3	9.2	90.0	
	2	2	75.4	9.9	88.8	
	3	3	78.3	12.5	88.2	

Table 13. Accuracy levels achieved at the different versions of weighted K-means clustering at the noisy Iris dataset; in each line, the accuracy at 2 noise features is on top, and at 4 noise features, on bottom; means, standard deviations and maxima are over a hundred of random initializations when applicable.

	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std dev	Max	
K-Means	2	0	67.1	6.4	76.7	
			66.7	7.0	80.0	
WK-Means	2	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	68.7			
			69.3			
iWK-Means	2	1.1	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 14: Accuracy of the K-Means versions at the Wine dataset with 7 and 13 noise features (top and bottom, respectively) ; means, standard deviations, maxima and minima are over a hundred of random initializations when applicable.

Algorithm	Exponent β at		Accuracy, %			
	Distance	Weight	Mean	Std	Max	Min
K-Means	2	0	93.0	6.5	96.6	52.3
			87.5	11.0	93.3	59.0
WK-Means	2	2.7	91.9	7.1	95.5	57.3
		2.6	89.4	10.6	94.9	50.0
MWK-Means	1.6	1.6	92.2	6.8	95.5	55.6
	1.4	1.4	88.3	10.6	94.4	55.1
iK-Means	2	0	94.4			
			93.3			
iWK-Means	2	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**			

*Optimal for the dataset with 7 extra noise features. ** Optimal for the dataset with 13 extra noise features.

Table 15. Average accuracy levels at GM 5 cluster 500×6 datasets with 2 noise features added.

Algorithm	Mean β	Std β	Average accuracy over 100 runs, %	Std	Average Max, over the datasets	Std of Max
K-Means	-	-	38.5	5.2936	47.62	6.4363
WK-Means	1.5	0.0816	56.56	5.2638	78.56	9.4482
MWK-Means	1.54	0.0699	60.26	6.3688	80.52	8.6970
iK-Means	-	-	-	-	37.7	6.8633
iWK-Means	2.14	1.0058	-	-	66.12	11.5774
iMWK-Means	1.79	0.4841	-	-	70.28	11.9090

Table 16. Average accuracy levels at GM 5 cluster 500×15 datasets with 10 noise features added.

Algorithm	Mean β	Std β	Average accuracy over 100 runs, %	Std	Average Max, over datasets	Std of Max
K-Means	-		0.4671	0.0957	0.6276	0.0886
WK-Means	3.44	1.5469	0.7244	0.0997	0.8984	0.0971
MWK-Means	1.4	0.0707	0.7882	0.0658	0.9336	0.0424
iK-Means	-	-	-	-	0.5584	0.1629
iWK-Means	3.18	1.4412	-	-	0.8076	0.0964
iMWK-Means	1.6	0.2739	-	-	0.8882	0.0925

Table 17. Average accuracy levels at GM 12-cluster 1000×25 datasets with 25 noise features added.

Algorithm	Mean β	Std β	Average accuracy over 100 runs, %	Std	Average Max, over datasets	Std Max
K-Means	-	-	0.2074	0.0184	0.2860	0.0379
WK-Means	4.76	0.2302	0.5205	0.0536	0.7136	0.0666
MWK-Means	1.32	0.0447	0.6843	0.0315	0.8414	0.0147
iK-Means	-	-	-	-	0.2192	0.0582

iWK-Means	2.88	1.359	-	-	0.3936	0.1447
iMWK-Means	1.48	0.1789	-	-	0.5488	0.0327

Table 18. Average accuracy levels at GM 12-cluster 1000×50 datasets with 25 noise features added.

Algorithm	Mean β	Std β	Average accuracy over 100 runs, %	Std	Average Max, over datasets	Std Max
K-Means	-	-	0.9279	0.0107	0.9998	0.0004
WK-Means	4.18	0.7981	0.8042	0.0108	0.9312	0.0062
MWK-Means	1.22	0.0447	0.8522	0.0818	0.9868	0.0290
iK-Means	-	-	-	-	0.8852	0.0815
iWK-Means	1.84	1.3372	-	-	0.3736	0.2541
iMWK-Means	1.9	0.4690	-	-	0.7076	0.0984

Table 19. Results of the experiments with the semi-supervised learning of the exponent β for WK-Means in datasets with noise features. The parenthesis at a data set name presents the original number of features plus the number of noise features. The means, modes and optima are taken over 50 runs of the semi-supervised learning algorithm (B option). The parentheses at Modal beta values present the frequencies of the modes.

Real World datasets	Exponent β			Accuracy, %	
	Mean	Modal	Optimal	Mean	Max
Iris (4+2)	1.23	1.2 (54%)	1.2	82.1/3.3	88.8
Iris (4+4)	1.21	1.2(64%)	1.2	85.4/2.5	89.4
Wine (13+7)	3.59	2.9(6%)	2.7	90.7/1.3	93.2
Wine (13+13)	3.74	3.1(10%)	2.6	85.4/1.8	89.1
Pima Indians (8+4)	1.55	1.5(22%)	1.9	63.2/1.6	65.3
Pima Indians (8+8)	1.74	1.3(16%)	1.7	65.1/1.48	66.6
Hepatitis (19+10)	2.10	1.0(58%)	1.0	77.3/1.9	78.8
Hepatitis (19+20)	2.65	1.0(36%)	1.5	77.6/1.3	80.1
Synthetic datasets*	Mean	Mean of Modal	Mean of Optima	Mean	Mean of Max
500x(6+2)	1.69/2.2	1.49/0.57	1.50	55.1/4.9	59.0/5.1
500x(15+10)	2.72/8.9	1.84/7.60	3.44	70.7/9.7	74.7/10.2
1000x(25+25)	4.30/1.6	4.46/3.36	4.76	50.8//5.3	53.9/5.1
1000x(50+25)	4.02/3.1	3.48/13.48	4.18	78.2/0.9	81.4/1.0

*A summary of all the experiments per Gaussian Model is presented. The number after the slash is the standard deviation computed over all the sets generated according to the corresponding Gaussian Model.

Table 20. Results of the experiments with the semi-supervised learning of the exponent β for iMWK-Means in datasets with noise features. The parenthesis at a data set name presents the original number of features plus the number of noise features. The means, modes and optima are taken over 50 runs of the semi-supervised learning algorithm (B option). The parentheses at Modal beta values present the frequencies of the modes.

Real World datasets	Exponent β			Accuracy, %	
	Mean	Modal	Optimal	Mean	Max
Iris (4+2)	1.08	1.1(52%)	1.1	94.8/1.6	96.0
Iris (4+4)	1.13	1.1(40%)	1.1	94.4/1.6	96.0
Wine (13+7)	1.60	1.1(32%)	2.2	93.7/1.7	95.5
Wine (13+13)	1.77	1.1(48%)	1.1	93.3/2.0	94.9
Pima Indians (8+4)	2.86	3.0(14%)	4.1	65.2/5.6	67.2
Pima Indians (8+8)	2.41	2.6(22%)	2.6	65.6/2.6	67.2
Hepatitis (19+10)	2.57	4.5(40%)	4.5	79.7/3.1	83.3
Hepatitis (19+20)	2.03	1.2(40%)	4.5	78.3/3.3	79.3
Synthetic datasets*	Mean	Mean of Modal	Mean of Optima	Mean	Mean of Max
500x(6+2)	1.74/3.0	1.71/4.9	1.79	69.4/12.4	70.3/11.9
500x(15+10)	1.58/2.9	1.52/2.4	1.6	88.3/9.7	88.8/9.2
1000x(25+25)	1.47/1.8	1.5/1.8	1.48	54.5/3.5	54.9/3.3
1000x(50+25)	1.82/3.5	1.88/4.8	1.90	70.4/10.8	70.8/9.84

*A summary of all the experiments per Gaussian Model is presented. The number after the slash is the standard deviation computed over all the sets generated according to the corresponding Gaussian Model.

Table 21. The total time, in seconds, taken by an algorithm to make a single run.

GM noisy 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

Figure 1: Wine dataset, on the left, and its version with 13 noise features, on the right, on the plane of the two first principal components, so that the axes correspond to the first and second singular vectors of the data matrix. The classes are shown by different shapes for data points.

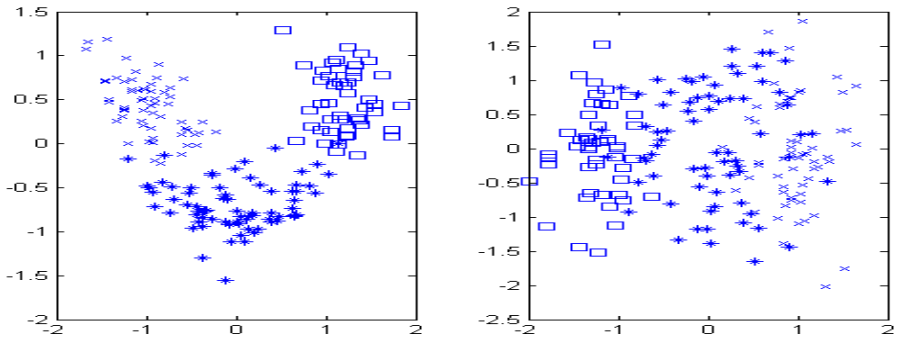

Figure 2: Illustration of one of the generated GM datasets, on the left, and its version with the noise, on the right. The axes correspond to the first and second singular vectors of the data matrix.