Pattern Recognition Letters
journal homepage: www.elsevier.com

Improving cluster recovery with feature rescaling factors

Renato Cordeiro de Amorim[a,**], Vladimir Makarenkov[b]

[a] School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park CO4 3SQ, UK.
[b] Département d'informatique, Universit é du Québec à Montréal, C.P. 8888 succ. Centre-Ville, Montreal(QC) H3C 3P8 Canada.

ABSTRACT

Clustering algorithms aim to create a partition of a data set so that each cluster contains homogeneous entities according to a certain criterion. The data pre-processing stage is crucial in clustering. Features may describe entities using different scales. To rectify this, one usually applies feature normalisation, aiming at rescaling features so that none of them overpowers the others in the objective function of the selected clustering algorithm. In this paper, we argue that the rescaling process should not treat all features identically. Instead, it should favour the features that are more meaningful for clustering. With this in mind, we introduce a feature rescaling method that takes into account the within-cluster degree of relevance of each feature. Our comprehensive simulation study, carried out on the data with and without noise features, clearly demonstrate that our novel data normalization approach allows our clustering algorithm to outperform the existing clustering methods that use traditional data normalization.

1. Introduction

The main aim of any clustering algorithm is to produce a set of clusters so that the entities belonging to a given cluster are homogeneous according to some criteria. Such algorithms have found use in a number of different fields such as bioinformatics, data mining, computer vision, etc. (Suzuki and Shimodaira, 2006; Panda et al., 2017; Berkhin, 2006; de Souto et al., 2008; de Amorim and Makarenkov, 2016; de Amorim et al., 2017).

Given a data set $X = \{x_1, x_2, ..., x_n\}$ in which $x_i \in \mathbb{R}^m$, a partitional crisp clustering algorithm aims to produce a clustering $S = \{S_1, S_2, ..., S_k\}$, such that $\left|\bigcup_{l=1}^{k} S_l\right| = n$ and $S_i \cap S_j = \emptyset$ for $i, j = 1, 2, ..., k$ and $i \neq j$. It is common for partitional clustering algorithms to consider a distance between entities, such as for example the squared Euclidean distance given by $d(x_i, x_j) = \sum_{v=1}^{m}(x_{iv} - x_{jv})^2$. There are indeed different clustering approaches, including crisp and fuzzy data partitioning as well as hierarchical clustering (Aggarwal and Reddy, 2014; Xu and Tian, 2015). Algorithms under the fuzzy clustering approach allow a given entity x_i to be assigned to more than one cluster, each assignment has a degree of membership and these usually add to one. Hierarchical clustering algorithms seek to produce a hierarchy of clusters, usually presented as a tree. Thus, an entity x_i may belong to more than one cluster when such clusters are at different levels of the tree. The approach presented in this paper relates to the crisp partitional clustering.

A special attention in clustering should be given to data pre-processing. During this stage, one of the main concerns is to apply the most appropriate data normalisation technique. The m features used to describe each entity $x_i \in X$ may be defined using different scales. Thus, a feature v with a wider scale than any other feature in X will have a higher contribution to clustering than any other individual feature. This may lead to poor cluster recovery, particularly if v is not as meaningful as the other features in X. With this in mind, data sets are usually normalised so that no feature overpowers others. The main data normalisation techniques used in data mining are the range normalisation, min-max normalisation, rescaling to unit length, z-scores and robust z-scores.

Having a balanced data set with features defined at the same scale is certainly a good starting point. However, in this paper we maintain that data normalisation should not aim to treat all features identically. It should instead aim to favour features that are more meaningful for clustering. This is the main contribution of our study. We show that the cluster-specific feature weights produced by a particular algorithm can be inter-

**Corresponding author
e-mail: r.amorim@essex.ac.uk (Renato Cordeiro de Amorim), makarenkov.vladimir@uqam.ca (Vladimir Makarenkov)

preted as feature rescaling factors. We also show that rescaling a data set using these features weights leads to a better cluster recovery. Our approach is quite unusual because our feature weights are cluster-specific. Thus, in our method each feature v is rescaled with k factors, where k is the number of clusters.

The remainder of the paper is organised as follows: Section 2 describes relevant related work in both clustering and feature rescaling; Section 3 presents our method in details, explaining in particular why it should work; Section 4 describes our experimental set up and the results we obtained; Section 5 finalises our paper by presenting the main contributions of this study.

2. Related work

We begin this section by briefly reviewing relevant clustering algorithms. We then discuss popular methods for feature rescaling.

2.1. Clustering algorithms

K-means (MacQueen et al., 1967) is arguably the most popular partitional clustering algorithm (Jain, 2010; Steinley, 2006). It aims to partition a data set X, containing n entities, into k non-overlapping clusters $S = \{S_1, S_2, ..., S_k\}$, so that $|\bigcup_{l=1}^{k} S_l| = n$. It does so by minimising the within-cluster sum of squares:

$$P(U, Z) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{v=1}^{m} u_{il}(x_{iv} - z_{lv})^2, \quad (1)$$

where u is a $n \times k$ binary matrix in which the value of u_{il} indicates whether or not $x_i \in S_l$, and z_l is the centroid of cluster S_l (i.e., its centre of gravity). If the square Euclidean distance is used in (1), it becomes straightforward that $z_{lv} = |S_l|^{-1} \sum_{i=1}^{n} u_{il} x_{iv}$. We can summarise k-means in three steps:

1. Select k entities from X uniformly at random and copy their values into the initial centroids $Z = \{z_1, z_2, ..., z_k\}$.
2. Assign each $x_i \in X$ to the cluster S_l whose centroid z_l is the nearest to x_i and update u_{il} accordingly. If this step produces no change in u, then stop.
3. Update z_l to the component-wise mean of $x_i \in S_l$, for $l = 1, 2, ..., k$.

The algorithm above is guaranteed to converge. This is true for two reasons: (i) the number of possible partitions may be large, but it is finite and (ii) the output of (1) is monotonically decreasing, and by consequence no clustering is repeated. However, there is no guarantee the final clustering will be optimal. Due to its greedy nature, the final clustering found by k-means is usually only a local minima solution. Moreover, this clustering is highly dependent on the initial centroids (usually set up at random). In fact, finding the optimal clustering minimising (1) is an NP-hard problem, even for $n = 2$ (Aloise et al., 2009).

There has been a considerable research effort aiming at designing algorithms capable of producing good initial centroids for k-means. Here, we discuss two methods that we find to be particularly relevant. We direct readers interested in a wider view to the following papers Yuan et al. (2004); Hatamlou (2012); Erisoglu et al. (2011); Sun et al. (2002); Steinley and Brusco (2007), and references therein.

K-means++ (Arthur and Vassilvitskii, 2007) is a very popular implementation of k-means, which has become the default k-means program in MATLAB. This algorithm selects the first centroid at random from the entities, and the others using a weighted probability related to the distances between entities and their closest centroid already chosen.

1. Set $l = 1$. Select an entity from X uniformly at random and copy its values to z_l.
2. Increment l by one. Select an entity x_j from X at random, with probability $\frac{D(x_j)^2}{\sum_{i=1}^{n} D(x_i)^2}$ and copy its values to z_l.
3. Repeat the steps above until $l = k$.
4. Run k-means using the $\{z_1, z_2, ..., z_k\}$ as initial centroids.

In the above, $D(x_i)$ represents the distance between x_i and its nearest centroid. Experiments show that k-means++ has a faster convergence to a lower criterion output (1) than the traditional k-means algorithm (Arthur and Vassilvitskii, 2007).

Intelligent k-means (ik-means) (Mirkin, 2012) is another popular algorithm designed to determine good initial centroids for k-means. It does so by using the concept of anomalous patterns. We describe the main steps of this algorithm below.

1. Find the entity $x_i \in X$ that is the farthest one from the data centre (z_c), and copy its values to z_t.
2. Run k-means on X with two initial centroids, z_t and z_c, leading to the clusters S_t and S_c. During this clustering, do not allow z_c to move at the centroid update step.
3. If $|S_t| > \theta$ add z_t to Z' and remove each $x_i \in S_t$ from X. If $|X| > 0$, then go to Step 1.
4. Run k-means on the whole original data set using the centroids in Z' as initial centroids.

The above identifies a centroid z_t and the related cluster S_t by iteratively minimising:

$$P(U, Z) = \sum_{i=1}^{n} \sum_{v=1}^{m} u_{it}(x_{iv} - z_{tv})^2 + \sum_{i=1}^{n} \sum_{v=1}^{m} u_{ic}(x_{iv} - z_{cv})^2. \quad (2)$$

Given that clustering is usually done after data normalisation leading to the data centre (z_c) of zero, we can rewrite (2) as follows:

$$P(U, Z) = \sum_{i=1}^{n} \sum_{v=1}^{m} u_{it}(x_{iv} - z_{tv})^2 + \sum_{i=1}^{n} \sum_{v=1}^{m} u_{ic} x_{iv}^2. \quad (3)$$

This anomalous pattern method identifies suitable initial centroids for k-means as well as the number of clusters k, and it does so quite successfully (Chiang and Mirkin, 2010). In this paper we are not interested in finding the number of clusters in a data set, so when using this initialisation we set $\theta = 0$ and select the k centroids in Z' with the largest cardinality.

The final clustering generated by k-means depends heavily on the initial centroids. Both k-means++ and ik-means attempt to identify good initial centroids, but unfortunately this is not the only weakness in k-means. The k-means criterion

(1) assumes that every feature in the data set is equally relevant, which is hardly the case in real-life scenarios. With this in mind de Amorim and Mirkin (2012) introduced the intelligent Minkowski weighted k-means (*imwk*-means) - an algorithm capable of successfully calculating within-cluster feature weights that improve cluster recovery (de Amorim, 2016; Melvin et al., 2016). The Minkowski distance between entity $x_i \in S_l$ and centroid z_l is defined by:

$$d(x_i, z_l) = \sum_{v}^{m} w_{lv}^{p} |x_{iv} - z_{lv}|^p, \tag{4}$$

where w_{lv} is the weight of feature v at cluster S_l, and p is a user-defined Minkowski exponent. This method applies the Minkowski distance rather than the Euclidean distance to avoid clusterings biased solely towards Gaussian (spherical) clusters. The Minkowski exponent p allows one to control the bias shape of clusters. For instance, at $p = 1$ clusters are biased towards diamond shapes, at $p = 2$ they are biased towards spherical shapes, and at $p \to \infty$ the bias is towards squares. Clearly, other values of p would lead to intermediary shapes to those stated above. Equation (4) is in fact the p^{th} power of the Minkowski distance, which is analogous to the use of the squared Euclidean distance in k-means. The distance (4) leads to the new optimization criterion:

$$P(U, Z, W) = \sum_{l=1}^{k} \sum_{i=1}^{n} \sum_{v=1}^{m} u_{il} w_{lv}^{p} |x_{iv} - z_{lv}|^p. \tag{5}$$

The minimisation of (5) subject to a crisp clustering and $\sum_{v=1}^{m} w_{lv} = 1$ for $l = 1, 2, ..., k$ implies:

$$w_{lv} = \frac{1}{\sum_{j=1}^{m} \left[\frac{D_{lv}}{D_{lj}} \right]^{\frac{1}{p-1}}}, \tag{6}$$

where $D_{lv} = \sum_{i=1}^{n} u_{il} |x_{iv} - z_{lv}|^p$. We can minimise (5) by adding an extra step to k-means. We refer to this as the Minkowski weighted k-means (*mwk*-means). The steps of this algorithm are as follows:

1. Select k entities from X uniformly at random, copy their values to the initial centroids $Z = \{z_1, z_2, ..., z_k\}$. Set each $w_{lv} = m^{-1}$.
2. Assign each $x_i \in X$ to the cluster S_l whose centroid z_l is the nearest to x_i as per (4), and update u_{il} accordingly. If this step produces no change in u, stop.
3. Update each z_l to the Minkowski centre of its cluster S_l (see below).
4. Update each w_{lv} as per (6). Go back to Step 2.

The Minkowski centre for feature v at cluster S_l is the value μ that minimises $\gamma_v(\mu) = \sum_{i=1}^{n} u_{il} |x_{iv} - \mu|^p$. Notice that at $p \geq 1$, $\gamma(\mu)$ is a U-shaped curve with a minimum in the interval $[min(x_v), max(x_v)]$. We can then minimise $\gamma(\mu)$ using standard methods for convex optimisation. For instance, to find the Minkowski centre for feature v at cluster S_l we can start with $\mu = |S_l|^{-1} \sum_{i=1}^{n} u_{il} x_{iv}$. We then move μ by a fixed amount (0.001, say) per step to the side that reduces γ_v. The *imwk*-means includes a Minkowski-based *ik*-means initialisation designed to find good initial centroids as well as good feature weights, as we can see below.

1. Set z_c to be the Minkowski centre of X, and each $w_{lv} = m^{-1}$.
2. Find the entity $x_i \in X$ that is the farthest from z_c using (4) and copy its values to z_t.
3. Run *mwk*-means using z_c and z_t as initial centroids, leading to the clusters S_c and S_t. In Step 3 of *mwk*-means do not allow z_c to move.
4. Add z_t to Z' and w to W'.
5. Remove all entities $x_i \in S_t$ from X. If $|X| > 0$ go to Step 2.
6. Keep in Z' and W' only the elements related to the k clusters with the highest cardinality.
7. Run *mwk*-means on the original data set X initialised with the centroids in Z' and weights in W'.

The *imwk*-means algorithm clearly supports the intuitive idea that a feature v may be more meaningful to one cluster than to another. We model this using w_{lv} to set the degree of relevance of feature v at cluster S_t.

2.2. Feature rescaling

Clustering algorithms usually require feature rescaling in the data pre-processing step. The general idea is to balance the values of features so that those with a higher scale do not overpower others. For instance, when clustering individuals the feature weight (measured in kilograms) will have a higher contribution to the criterion (1) than the feature height (measured in meters). This happens because in absolute values the weight of humans tends to be considerably larger than their height (in other words, the variance of weight is larger than that of height). Once all features present values on a common scale, the data set can be clustered.

There are different feature rescaling methods that can be applied during the data pre-processing step of clustering. Here, we focus on the most popular of them.

Z-scores

The z-score normalisation is arguably the most popular approach of feature rescaling. The z-score of x_{iv} is given by:

$$x'_{iv} = \frac{x_{iv} - \bar{x}_v}{\sigma_v}, \tag{7}$$

where $\bar{x}_v$ and σ_v are the mean and standard deviation of feature v, respectively. The standardised x'_{iv} represents the number of standard deviations by which the original x_{iv} is above $\bar{x}_v$. A popular extension of this method is the robust z-score normalisation in which the mean is replaced by the median and the standard deviation by MAD (Median Absolute Deviation). This method is more robust than z-scores in the presence of outliers.

Range normalisation

The range normalisation is a popular alternative to the z-score, particularly in cluster analysis . The normalised value of x_{iv} is computed as follows:

$$x'_{iv} = \frac{x_{iv} - \bar{x}_v}{max(x_v) - min(x_v)}, \tag{8}$$

where $max(x_v)$ and $min(x_v)$ are the maximum and minimum values of feature v, respectively. There is a crucial difference between the range normalisation and z-scores, the latter is biased toward unimodal distributions. This is probably easier to explain with an example. Let us take two features, a unimodal v_1 and a multimodal v_2. The standard deviation of v_2 is likely to be higher than that of v_1. Thus, the z-score value of v_1 will be higher than that of v_2 even though v_2 has a better cluster information.

Min-max normalisation
This is arguably the simplest method one can use to normalise features. It rescales the features of a given data set to the interval $[0, 1]$:

$$x'_{iv} = \frac{x_{iv} - min(x_v)}{max(x_v) - min(x_v)}. \quad (9)$$

Rescaling to unit length
A feature v can also be interpreted as being a vector. Thus, it is possible to normalise the components of v so that this vector has a length of one. In this case, the normalised value of x_{iv} is obtained as follows:

$$x'_{iv} = \frac{x_{iv}}{\|x\|}, \quad (10)$$

where $\|x_v\| = \sqrt{\sum_{i=1}^{n} x_{iv}^2}$ is the Euclidean length of feature v.

3. Clustering with feature rescaling factors

Feature rescaling methods such as those discussed in Section 2.2 are certainly a good starting point, but we maintain their application cannot be the final step of the data pre-processing stage. This stage should not aim at treating all features equally, but should instead favour features that have a higher degree of relevance for clustering. With this in mind, let us analyse the weights generated by *imwk*-means. Our objective here is to get a set of weights minimising (5) subject to $\sum_{v=1}^{m} w_{lv} = 1$ for $l = 1, 2, ..., k$ within a crisp clustering criterion (i.e., $S_i \cap S_j$ for $i, j = 1, 2, ..., k$ and $i \neq j$). Given that $D_{lv} = \sum_{i=1}^{n} u_{il}|x_{iv} - z_{lv}|^p$, we can rewrite the function to be minimised (5) as follows:

$$P(U, Z, W) = \sum_{v=1}^{m} \sum_{l=1}^{k} w_{lv}^p D_{lv}.$$

Since we calculate feature weights one cluster at a time and $\sum_{v=1}^{m} w_{lv} = 1$ for $l = 1, 2, ..., k$, the following Lagrangian function can be formulated:

$$\mathcal{L}(W, \lambda) = \sum_{v=1}^{m} w_{lv}^p D_{lv} + \lambda \left(1 - \sum_{v=1}^{m} w_{lv}\right).$$

The partial derivatives of $\mathcal{L}$ with respect to w_{lv} and λ can be equated to 0. They are as follows:

$$\frac{\partial \mathcal{L}}{\partial w_{lv}} = p w_{lv}^{p-1} D_{lv} - \lambda = 0, \quad (11)$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_{v=1}^{m} w_{lv} = 0, \quad (12)$$

respectively. Equation (11) leads to:

$$w_{lv} = \left(\frac{\lambda}{p D_{lv}}\right)^{\frac{1}{p-1}}. \quad (13)$$

Substituting (13) into (12), we obtain:

$$\sum_{v=1}^{m} \left(\frac{\lambda}{p D_{lv}}\right)^{\frac{1}{p-1}} = 1,$$

leading to:

$$(\lambda)^{\frac{1}{p-1}} = \frac{1}{\sum_{v=1}^{m} \left(\frac{1}{p D_{lv}}\right)^{\frac{1}{p-1}}}.$$

and by consequence to Equation (6). The above demonstrates that the features generated by *imwk*-means are in fact quite specific: they minimise (5) and model the within-cluster degree of relevance of each feature. Thus, they can be used to rescale a data set in a rather unconventional way. Given u and feature weights w, we can rescale a data set by setting:

$$x'_{iv} = \sum_{l=1}^{k} u_{il} x_{iv} w_{lv}. \quad (14)$$

This is unconventional because a given feature v is rescaled with k different factors $w_1, w_2, ..., w_k$. We propose to improve cluster recovery with feature rescaling factors following the steps below.

1. Standardise the data set X using either (7), (8), (9), or (10).
2. Find a clustering U and weights W by applying the *imwk*-means to the standardised data set.
3. Rescale X using U and W by applying (14).
4. Apply *imwk*-means to the rescaled data set, leading to the clustering U'.

One should note that rescaling a feature with (14) only makes sense because the weights in W minimise (5). The *imwk*-means requires a user-defined parameter p and its rescaled version requires two values of p. The value of p_1 is used to generate the clustering U and weights W (Step 2) and the value p_2 is used in clustering of the rescaled data set (Step 4). Of course, one could set $p_1 = p_2$.

According to Formula (14), the rescaling part of the algorithm has a linear time complexity with respect to the number of clusters k, the number of features m, and the number of entities n. Thus, our rescaling would not increase the asymptotic time complexity of the k-means or *imwk*-means algorithms.

4. Experimental results

The data pre-processing stage is crucial for any clustering algorithm. In this step, features are usually put on the same scale so that none overpowers any of the others. The main objective of our experiments is to demonstrate that a rescaling favouring meaningful features leads to better cluster recovery. This happens because the feature rescaling factors generated by our method minimise sum of within cluster distances (see Section

3). Hence, we envisage that our method could be used in the data pre-processing step of any k-means-based clustering algorithm.

In this section, we experiment with three algorithms: (i) k-means++, giving the reader a clear idea of a generally accepted baseline for each data set; (ii) *imwk*-means, allowing us to generate the feature re-scaling factors; (iii) *imwk*-means with a data set rescaled using our method. We show that our rescaling method improves cluster recovery. We could have applied our method with other clustering algorithms (including k-means++), but decided not to do so here because this would necessarily increase the length of our paper.

We divided the remainder of this section into two parts. First, we explain the details related to the data sets we have used in our experiments. Then, we present and discuss the obtained results.

4.1. Experimental setup

In this paper, we experimented with a total of 600 synthetic data sets generated under 12 different configurations (see Table 1), with and without added noise. These data sets contain spherical Gaussian clusters. Their covariance matrices are diagonal, with the same diagonal value σ^2, generated at each cluster randomly between 0.5 and 1.5. Each centroid component was generated independently from a Gaussian distribution with zero mean and unity variance. Each cluster has a cardinality taken from a uniformly random distribution, subject to a minimum of 20 entities. We initially generated 50 data sets under each of the following configurations: (i) 1000 entities over six features partitioned into three clusters (1000x6-3); (ii) 1000 entities over 12 features partitioned into six clusters (1000x12-6); (iii) 1000 entities over 20 features partitioned into ten clusters (1000x20-10).

For each data set containing m features, we have generated two data sets including $\lceil \frac{m}{2} \rceil$ noise features (leading to a total of $\lceil 1.5m \rceil$ features) and one other data set with within cluster noise. Here, we experiment with three models of noise. In the first, we considered a noise feature (NF) as a feature containing solely uniformly random values. In the second, we considered a noise feature as one containing random values from a Gaussian distribution (NNF). In our third noise model we select 50% of the $m \times k$ feature segments uniformly at random, and then substitute the selected segments with uniformly random values - creating within cluster noise (WCN). This approach has quadrupled the number of data sets tested in our experiments. In all of our experiments we have normalised the features in a data set using one of the methods described in Section 2.2.

Given that we know the true labels for all our data sets (ie. their true structure is known), it makes sense to measure cluster recovery using an external validation index. In the case of clustering, the strongest contender is the Adjusted Rand Index (ARI) (Rand, 1971). The ARI between the clusterings $S = \{S_1, S_2, ..., S_k\}$ and $U = \{U_1, U_2, ..., U_r\}$ is defined as folows:

$$ARI(S,U) = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{n}{2}}, \quad (15)$$

Table 1: There are 50 data sets for each of the configurations below. Each data set contains Gaussian clusters with different spreads and cardinalities. We added noise features containing uniformly random values (NF), normal random values (NNF), or within cluster noise (WCN) to the data sets in some configurations.

	Noise (%)	Features				Clusters (k)
		normal noise	uniform noise	original	total (m)	
1000x6-3	0.00	0	0	6	6	3
1000x12-6	0.00	0	0	12	12	6
1000x20-10	0.00	0	0	20	20	10
1000x6-3 +3NF	33.33	0	3	6	9	3
1000x12-6 +6NF	33.33	0	6	12	18	6
1000x20-10 +10NF	33.33	0	10	20	30	10
1000x6-3 +3NNF	33.33	3	0	6	9	3
1000x12-6 +6NNF	33.33	6	0	12	18	6
1000x20-10 +10NNF	33.33	10	0	20	30	10
1000x6-3 WCN	50.00	0	0	6	6	3
1000x12-6 WCN	50.00	0	0	12	12	6
1000x20-10 WCN	50.00	0	0	20	20	10

where $n_{ij} = |S_i \cap U_j|$, $a_i = \sum_{j=1}^{r} |S_i \cap U_j|$, $b_j = \sum_{i=1}^{k} |S_i \cap U_j|$.

4.2. Results and analysis

In our first set of experiments, we compared the four feature rescalings presented in Section 2.2 using the data sets described in Section 4.1. Table 2 shows the average ARI between the clusterings generated by k-means++ using these four feature rescaling approaches. Given that k-means usually returns different clustering solutions for different random initial partitions, we ran k-means++ 100 times per data set (notice that we tested 50 different data sets per parameter configuration). Some interesting patterns can be found when observing the results reported in Table 2. For example, the range normalisation produces slightly better results than the others in the original data sets (no noise features) as well as in the data sets containing normal noise values (NNF). However, in the data sets containing uniformly random noise features (NF) or within cluster noise (WCN) it is the z-score normalisation that produces the best results, while the range-based normalisations (min-max and range normalisation) perform poorly.

The main reason for this is that the features containing uniformly random values have no cluster structure. Thus, the standard deviation of such noise features is likely to be higher than that of original features. Given that z-score (7) include a division by the standard deviation, uniformly random noise features will have lower standard values. By consequence, these noise features will have a lower contribution to the clustering (1).

Given the results in Table 2, we decided to study in more details the range and z-score normalisations. We ran two sets of experiments. In the first of them, our aim was to find out whether there are parameters for the rescaled $imwk - means$ that would lead to better cluster recovery than the best possible clustering by *imwk*-means, and the expected k-means++ clustering. As k-means++ is among the most popular variations of k-means (Arthur and Vassilvitskii, 2007), it would be important to propose a method that outperforms it, regardless of the data normalization being used.

Table 3 presents the results of our experiments, obtained for data sets with and without noise, all normalised using the range

Table 2: Average ARI and standard deviation values for the results found by k-means++. We ran k-means++ 100 times per data set. There are 50 data sets per parameter configuration. Each of the four main columns presents the results for a different normalisation approach.

	min-max		range norm		z-score		unit length	
	ARI	std	ARI	std	ARI	std	ARI	std
1000x6-3	0.5145	0.22	0.5198	0.22	0.5060	0.21	0.4821	0.23
1000x12-6	0.6338	0.18	0.6356	0.18	0.6336	0.17	0.6222	0.18
1000x20-10	0.7680	0.12	0.7703	0.12	0.7708	0.11	0.7712	0.12
1000x6-3 +3NF	0.0365	0.11	0.0371	0.11	0.4550	0.21	0.4014	0.22
1000x12-6 +6NF	0.0994	0.11	0.0997	0.11	0.5820	0.17	0.5513	0.18
1000x20-10 +10NF	0.1706	0.10	0.1708	0.10	0.7233	0.14	0.7184	0.12
1000x6-3 +3NNF	0.4735	0.23	0.4748	0.23	0.4690	0.21	0.4102	0.22
1000x12-6 +6NNF	0.5837	0.17	0.5852	0.17	0.5840	0.17	0.5552	0.18
1000x20-10 +10NNF	0.7277	0.13	0.7286	0.13	0.7278	0.12	0.7205	0.12
1000x6-3 WCN	0.0594	0.06	0.0597	0.06	0.1645	0.09	0.1473	0.09
1000x12-6 WCN	0.0926	0.06	0.0920	0.06	0.1835	0.07	0.1767	0.07
1000x20-10 WCN	0.1051	0.03	0.1049	0.03	0.1735	0.05	0.1698	0.05

Table 3: A comparison between the ARI values for *imwk*-means and its rescaled version, supplied with good values of the exponents p_1 and p_2, and the expected ARI given by k-means++. The presented results are the averages over 50 data sets for each of the configurations below, each normalised using the range normalisation.

	k-means++		*imwk*-means		Rescaled *imwk*-means	
	ARI	std	ARI	std	ARI	std
1000x6-3	0.5198	0.224	0.5794	0.223	0.6474	0.191
1000x12-6	0.6356	0.177	0.7376	0.174	0.7958	0.136
1000x20-10	0.7703	0.121	0.9070	0.076	0.9390	0.055
1000x6-3 +3NF	0.0371	0.112	0.5541	0.283	0.6781	0.198
1000x12-6 +6NF	0.0997	0.111	0.7543	0.156	0.8307	0.116
1000x20-10 +10NF	0.1708	0.099	0.8239	0.082	0.9356	0.041
1000x6-3 +3NNF	0.4748	0.225	0.5864	0.215	0.6495	0.200
1000x12-6 +6NNF	0.5852	0.172	0.7358	0.174	0.7832	0.156
1000x20-10 +10NNF	0.7286	0.125	0.9050	0.059	0.9381	0.041
1000x6-3 WCN	0.0597	0.059	0.3916	0.194	0.4597	0.162
1000x12-6 WCN	0.0920	0.062	0.6669	0.157	0.6811	0.139
1000x20-10 WCN	0.1049	0.032	0.8481	0.049	0.8868	0.048

Table 4: A comparison between the ARI values for *imwk*-means and its rescaled version, supplied with good values of the exponents p_1 and p_2, and the expected ARI given by k-means++. The presented results are the averages over 50 data sets for each of the configurations below, each normalised using z-scores.

	k-means++		*imwk*-means		Rescaled *imwk*-means	
	ARI	std	ARI	std	ARI	std
1000x6-3	0.5060	0.214	0.5888	0.218	0.6617	0.189
1000x12-6	0.6336	0.173	0.7412	0.173	0.7996	0.129
1000x20-10	0.7708	0.112	0.8981	0.076	0.9376	0.053
1000x6-3 +3NF	0.4550	0.211	0.5793	0.214	0.6595	0.195
1000x12-6 +6NF	0.5820	0.171	0.7285	0.178	0.8138	0.127
1000x20-10 +10NF	0.7233	0.125	0.9024	0.067	0.9400	0.040
1000x6-3 +3NNF	0.4690	0.212	0.5794	0.226	0.6513	0.192
1000x12-6 +6NNF	0.5840	0.168	0.7381	0.179	0.7879	0.138
1000x20-10 +10NNF	0.7278	0.123	0.8946	0.066	0.9358	0.045
1000x6-3 WCN	0.1645	0.095	0.3635	0.180	0.4144	0.171
1000x12-6 WCN	0.1835	0.070	0.6128	0.169	0.6256	0.167
1000x20-10 WCN	0.1735	0.048	0.8386	0.068	0.8405	0.072

Table 5: A comparison between the ARI values for *imwk*-means and its rescaled version, supplied with parameters that work well on average, and the expected ARI given by k-means++. The presented results are the averages over 50 data sets for each of the configurations below, each normalised using the range normalisation.

	k-means++		*imwk*-means			Rescaled *imwk*-means			
	ARI	std	ARI	std	p	ARI	std	p_1	p_2
1000x6-3	0.5198	0.224	0.5249	0.227	2.8	0.5453	0.227	4.4	2.9
1000x12-6	0.6356	0.177	0.6434	0.191	2.5	0.6601	0.187	3.9	2.5
1000x20-10	0.7703	0.121	0.8294	0.116	2.5	0.8539	0.100	5.0	2.1
1000x6-3 +3NF	0.0371	0.112	0.4385	0.308	1.5	0.4622	0.307	1.4	2.8
1000x12-6 +6NF	0.0997	0.112	0.6820	0.190	1.6	0.7152	0.195	1.7	2.4
1000x20-10 +10NF	0.1708	0.099	0.7519	0.105	1.7	0.8619	0.070	2.0	1.7
1000x6-3 +3NNF	0.4748	0.225	0.5236	0.225	2.4	0.5341	0.219	4.9	2.6
1000x12-6 +6NNF	0.5852	0.172	0.6539	0.175	2.0	0.6518	0.183	4.7	2.4
1000x20-10+10NNF	0.7286	0.125	0.8286	0.103	2.5	0.8622	0.090	4.8	2.1
1000x6-3 WCN	0.0597	0.059	0.2524	0.208	1.6	0.2798	0.215	4.7	1.5
1000x12-6 WCN	0.0920	0.062	0.5677	0.191	1.5	0.5791	0.192	1.5	2.9
1000x20-10 WCN	0.1049	0.032	0.7594	0.116	1.6	0.7690	0.107	5.0	1.7

normalisation. There are 50 data sets for each configuration, this is the reason why we present the standard deviations for all the three competing algorithms. In this table we can see a clear pattern. The average ARI given by the rescaled *imwk*-means in this experiment is higher than that of *imwk*-means and k-means++. This pattern becomes even clearer as the number of features and clusters increases. It is also interesting to see that the standard deviation of the results obtained by the rescaled *imwk*-means is slightly lower than those of *imwk*-means and k-means++ in the majority of cases. Unsurprisingly the cluster recovery improvements provided by *imwk*-means and its rescaled version are higher in data sets containing noise features. This is a fair expectation for feature weighting algorithms. It is interesting to see that for *imwk*-means and its rescaled version we also have the following trend: the higher is the ARI, the lower is the standard deviation.

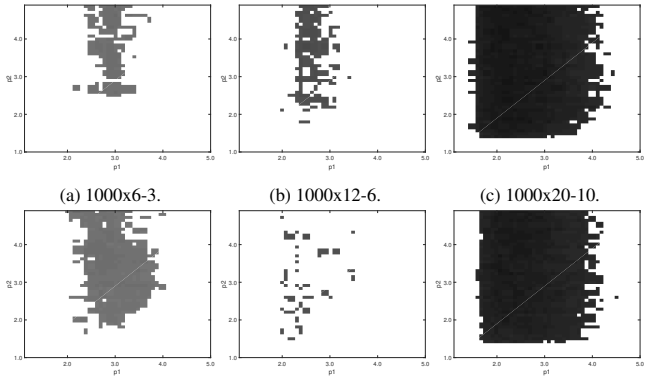
Table 4 reports the results for the experiments in which the data were normalised using z-scores. The general patterns are still the same. The rescaled *imwk*-means provides better results than *imwk*-means and k-means++. The only major difference is that now k-means++ produces better results for the data sets containing noise features composed of uniformly random values (NF). We have explained the reason for this in the beginning of this section. The k-means++ results for the data sets containing within cluster noise (WCN) are also somewhat better, but still poor overall.

In our second set of experiments our aim was to determine whether there is a pattern for suitable parameters of the rescaled *imwk*-means. In this scenario the first question one usually would ask is whether particular pairs of the exponent parameters p_1 and p_2 work well on average (over the 50 data sets for each configuration). Table 5 shows the results of these experiments for the data sets normalised using the range normalisation. The differences in ARI are not as large as before, but we can still see that the rescaled *imwk*-means is competitive and usually outperforms *imwk*-means. Table 6 presents the results for data sets normalised using z-scores. In this case, the rescaled *imwk*-means is still superior to *imwk*-means and k-means++.

Given the difficulty of finding clear patterns of good values for p_1 and p_2, we generated various figures showing the average ARI per pair (p_1, p_2) for each data set considered in our simulations. For easy comparison, we set to white each pixel representing a pair (p_1, p_2) which did not outperform k-means++. Notice that we have experimented with values of p from 1.1 to 5.0 (in steps of 0.1).

Table 6: A comparison between the ARI values for *imwk*-means and its rescaled version, supplied with parameters that work well on average, and the expected ARI given by *k*-means++. The presented results are the averages over 50 data sets for each of the configurations below, each normalised using *z*-scores.

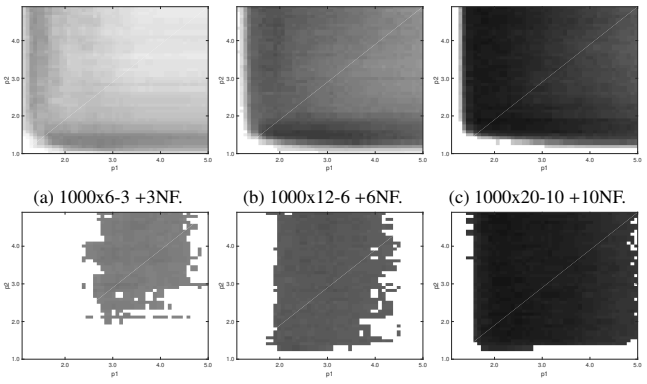
	k-means++		*imwk*-means			Rescaled *imwk*-means			
	ARI	std	ARI	std	p	ARI	std	p_1	p_2
1000x6-3	0.5060	0.214	0.5286	0.221	2.7	0.5406	0.229	3.9	2.6
1000x12-6	0.6336	0.173	0.6533	0.190	3.0	0.6463	0.184	4.3	2.5
1000x20-10	0.7708	0.112	0.8299	0.104	2.1	0.8489	0.110	3.8	2.1
1000x6-3 +3NF	0.4550	0.211	0.4771	0.200	3.8	0.5143	0.239	3.1	3.1
1000x12-6 +6NF	0.5820	0.171	0.6281	0.193	2.6	0.6496	0.205	5.0	3.0
1000x20-10 +10NF	0.7233	0.125	0.8221	0.110	2.5	0.8567	0.104	3.2	2.0
1000x6-3 +3NNF	0.4690	0.212	0.5072	0.232	2.4	0.5193	0.229	4.9	2.7
1000x12-6 +6NNF	0.5840	0.168	0.6577	0.186	1.8	0.6430	0.186	4.8	2.4
1000x20-10 +10NNF	0.7278	0.123	0.8282	0.084	2.4	0.8524	0.096	4.1	2.2
1000x6-3 WCN	0.1645	0.095	0.2466	0.189	2.0	0.2672	0.154	3.4	2.6
1000x12-6 WCN	0.1835	0.070	0.5199	0.204	1.6	0.5529	0.209	1.4	2.2
1000x20-10 WCN	0.1735	0.048	0.7496	0.115	1.6	0.7682	0.143	1.4	2.1



(a) 1000x6-3. (b) 1000x12-6. (c) 1000x20-10.

(d) 1000x6-3. (e) 1000x12-6. (f) 1000x20-10.

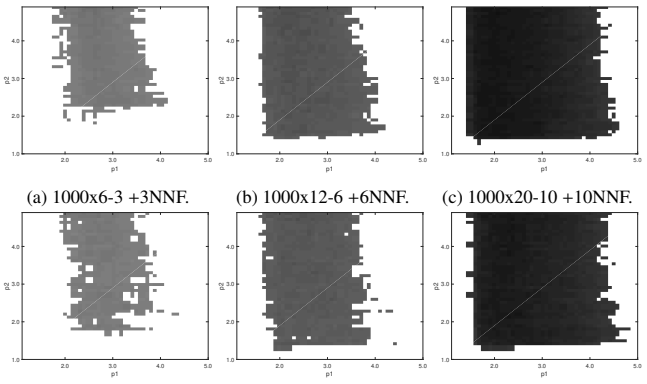
Fig. 1: Average ARI given by the rescaled *imwk*-means for each pair of the exponent parameters p_1 and p_2. No noise features were added to the data. White pixels represent pairs (p_1, p_2) which did not outperform *k*-means++. We applied the range normalisation to the data sets in Figures 1a, 1b and 1c. We applied *z*-scores to the data sets in Figures 1d, 1e and 1f.



(a) 1000x6-3 +3NF. (b) 1000x12-6 +6NF. (c) 1000x20-10 +10NF.

(d) 1000x6-3 +3NF. (e) 1000x12-6 +6NF. (f) 1000x20-10 +10NF.

Fig. 2: Average ARI given by the rescaled *imwk*-means for each pair of the exponent parameters p_1 and p_2. White pixels represent pairs (p_1, p_2) which did not outperform *k*-means++. The noise features (NF) are composed of uniformly random variables. We applied the range normalisation to the data sets in Figures 2a, 2b and 2c. We applied *z*-scores to the data sets in Figures 2d, 2e and 2f.



(a) 1000x6-3 +3NNF. (b) 1000x12-6 +6NNF. (c) 1000x20-10 +10NNF.

(d) 1000x6-3 +3NNF. (e) 1000x12-6 +6NNF. (f) 1000x20-10 +10NNF.

Fig. 3: Average ARI given by the rescaled *imwk*-means for each pair of the exponent parameters p_1 and p_2. White pixels represent pairs (p_1, p_2) which did not outperform *k*-means++. The noise features are composed of normally distributed random variables (NNF). We applied range the normalisation to the data sets in Figures 2a, 2b and 2c. We applied *z*-scores to the data sets in Figures 2d, 2e and 2f.

We begin our analysis describing the experiments with noise-free data sets. Figure 1 shows the ARI results for the experiments with these data sets normalised using the range normalisation and *z*-scores. The presented results indicate that in both cases (range normalisation and *z*-score) there is a limited number of pairs (p_1, p_2) that lead to high values of ARI for the data configurations 1000x6-3 and 1000x12-6. However, when the number of features and clusters increase (and by consequence the difficulty of producing a good clustering), we can observe that the majority of the exponent parameters (p_1, p_2) lead to high values of ARI. For instance, for the data configuration 1000x20-10 nearly all pairs (p_1, p_2), such that p_1 is located in the interval $[2, 4]$ lead to a high ARI.

Figure 2 shows an even more favourably pattern for the proposed rescaled *imwk*-means. These experiments concern data sets to which we have added noise features containing uniformly random values. When the data sets have been normalised used the range normalisation, nearly all possible pairs (p_1, p_2) led to a considerably high value of ARI. In the case in which data sets have been normalised with *z*-scores, we can clearly see that the more complex a data set is, the larger is the pool of pairs (p_1, p_2) producing a high value of ARI. In the case of the data sets under the configuration 1000x20-10 +10NF, nearly all pairs lead to a high value of ARI.

Figure 3 shows the same pattern. It illustrates the results of the experiments conducted with data sets to which we have added noise features containing Gaussian random values. Still, the more complex a data set is, the larger is the number of pairs (p_1, p_2) producing high values of ARI. We can also see that the difference between the ARI values generated by the rescaled *imwk*-means and *k*-means++ becomes larger. Again, in the case of the data configuration 1000x20-10 +10NNF, nearly all pairs of the exponent parameters lead to a high value of ARI.

Overall our experiments show that rescaling a data set using our method improves cluster recovery. This is hardly surprising given that our feature rescaling factors minimise the within-cluster sum of distances (see Section 3). Thus, our method can certainly be used during the data pre-processing stage of any clustering algorithm based on distance measures. Of course, one could ask why our method improves the cluster recovery of algorithms capable of applying feature weighting (eg. *imwk*-means). This happens because such algorithms start with a sub-

optimal set of weights, sometimes containing even random values (for details see the recent surveys de Amorim (2016); Deng et al. (2016); Kriegel et al. (2009, 2012) and references there in). Usually, these weights are then improved at each iteration. Our rescaling method leads to more compact clusters, so the effect of the final set of weights (the most improved) produced by *imwk*-means can be experienced from the first iteration.

5. Conclusion

Feature rescaling is a crucial part of the data pre-processing step in clustering. Typically original features describe entities located at different scales. Rescaling aims at balancing such features so that none of them overpowers the others in the objective function of the selected clustering algorithm. Here, we highlight that feature rescaling should in fact favour more meaningful features, rather than simply put all of them on the same scale.

Thus, we introduced a data rescaling method based on the *imwk*-means algorithm. The latter analyses a normalised data set and produces a set of cluster-based feature weights. It is indeed intuitive that some features should be more relevant to certain clusters than the others. We showed how these weights can be used to account for the degree of relevance of any given feature at a particular cluster. These cluster-based weights are then used as feature rescaling factors. Our rescaling approach is quite different from the classical ones because a given feature will be rescaled using k different factors, where k is the number of clusters. Our rescaling method can be used in the data pre-processing step of any distance-based clustering algorithm such as k-means, k-means++, *imwk*-means, etc.

Our approach works because the feature weights minimise our clustering criteria (5). Rescaling a data set using these weights as feature rescaling factors leads to more compact clusters, which are by consequence easier to be identified by a clustering algorithm. We demonstrated that our data pre-processing method generally produces a better cluster recovery than the existing methods in a series of simulations involving 600 synthetic data sets. These simulations were carried out with three types of noise features. First, we considered noise features containing uniformly random values. Second, we considered noise features containing Gaussian random values. Third, we considered features containing within cluster noise. Our experiments clearly demonstrated that the presented rescaling technique is effective and can be recommended for use as a data pre-processing step in clustering.

We have three directions for future research, (i) it would be interesting to investigate how the proposed feature rescaling method could be used as a data pre-processing step in the framework of supervised and semi-supervised machine learning approaches; (ii) we intend to investigate how our method behaves under other noise conditions; (iii) we are also interested in extending our method so that it is capable of dealing with data sets containing a very high number of features.

References

Aggarwal, C.C., Reddy, C.K., 2014. Data clustering. Chapman and Hall/CRC.

Aloise, D., Deshpande, A., Hansen, P., Popat, P., 2009. Np-hardness of euclidean sum-of-squares clustering. Machine learning 75, 245–248.

de Amorim, R.C., 2016. A survey on feature weighting based k-means algorithms. Journal of Classification 33, 210–242.

de Amorim, R.C., Makarenkov, V., 2016. Applying subclustering and lp distance in weighted k-means with distributed centroids. Neurocomputing 173, 700–707.

de Amorim, R.C., Mirkin, B., 2012. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. Pattern Recognition 45, 1061–1075.

de Amorim, R.C., Shestakov, A., Mirkin, B., Makarenkov, V., 2017. The minkowski central partition as a pointer to a suitable distance exponent and consensus partitioning. Pattern Recognition 67, 62–72.

Arthur, D., Vassilvitskii, S., 2007. k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics. pp. 1027–1035.

Berkhin, P., 2006. A survey of clustering data mining techniques, in: Grouping multidimensional data. Springer, pp. 25–71.

Chiang, M.M.T., 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, 3–40.

Deng, Z., Choi, K.S., Jiang, Y., Wang, J., Wang, S., 2016. A survey on soft subspace clustering. Information sciences 348, 84–106.

Erisoglu, M., Calis, N., Sakallioglu, S., 2011. A new algorithm for initial cluster centers in k-means algorithm. Pattern Recognition Letters 32, 1701–1705.

Hatamlou, A., 2012. In search of optimal centroids on data clustering using a binary search algorithm. Pattern Recognition Letters 33, 1756–1760.

Hubert, L., Arabie, P., 1985. Comparing partitions. Journal of classification 2, 193–218.

Jain, A.K., 2010. Data clustering: 50 years beyond k-means. Pattern recognition letters 31, 651–666.

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 (TKDD) 3, 1.

Kriegel, H.P., Kröger, P., Zimek, A., 2012. Subspace clustering. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2, 351–364.

MacQueen, J., et al., 1967. Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA.. pp. 281–297.

Melvin, R.L., Godwin, R.C., Xiao, J., Thompson, W.G., Berenhaut, K.S., Salsbury Jr, F.R., 2016. Uncovering large-scale conformational change in molecular dynamics without prior knowledge. Journal of chemical theory and computation 12, 6130–6146.

Mirkin, B., 2012. Clustering: a data recovery approach. CRC Press.

Panda, R., Mithun, N.C., Roy-Chowdhury, A.K., 2017. Diversity-aware multi-video summarization. IEEE Transactions on Image Processing 26, 4712–4724.

Rand, W.M., 1971. Objective criteria for the evaluation of clustering methods. Journal of the American Statistical association 66, 846–850.

de Souto, M.C., Costa, I.G., de Araujo, D.S., Ludermir, T.B., Schliep, A., 2008. Clustering cancer gene expression data: a comparative study. BMC bioinformatics 9, 497.

Steinley, D., 2004. Properties of the hubert-arable adjusted rand index. Psychological methods 9, 386.

Steinley, D., 2006. K-means clustering: a half-century synthesis. British Journal of Mathematical and Statistical Psychology 59, 1–34.

Steinley, D., Brusco, M.J., 2007. Initializing k-means batch clustering: A critical evaluation of several techniques. Journal of Classification 24, 99–121.

Sun, Y., Zhu, Q., Chen, Z., 2002. An iterative initial-points refinement algorithm for categorical data clustering. Pattern Recognition Letters 23, 875–884.

Suzuki, R., Shimodaira, H., 2006. Pvclust: an r package for assessing the uncertainty in hierarchical clustering. Bioinformatics 22, 1540–1542.

Xu, D., Tian, Y., 2015. A comprehensive survey of clustering algorithms. Annals of Data Science 2, 165–193.

Yuan, F., Meng, Z.H., Zhang, H.X., Dong, C.R., 2004. A new algorithm to get the initial centroids, in: Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, IEEE. pp. 1191–1193.