Neurocomputing 553 (2023) 126547

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

On k-means iterations and Gaussian clusters

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

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

ARTICLE INFO

Communicated by Zidong Wang

Keywords:
k-means
Clustering
Feature selection

ABSTRACT

Nowadays, k-means remains arguably the most popular clustering algorithm (Jain, 2010; Vouros et al., 2021). Two of its main properties are simplicity and speed in practice. Here, our main claim is that the average number of iterations k-means takes to converge ($\overline{\tau}$) is in fact very informative. We find this to be particularly interesting because $\overline{\tau}$ is always known when applying k-means but has never been, to our knowledge, used in the data analysis process. By experimenting with Gaussian clusters, we show that $\overline{\tau}$ is related to the structure of a data set under study. Data sets containing Gaussian clusters have a much lower $\overline{\tau}$ than those containing uniformly random data. In fact, we go considerably further and demonstrate a pattern of inverse correlation between $\overline{\tau}$ and the clustering quality. We illustrate the importance of our findings through two practical applications. First, we describe the cases in which $\overline{\tau}$ can be effectively used to identify irrelevant features present in a given data set or be used to improve the results of existing feature selection algorithms. Second, we show that there is a strong relationship between $\overline{\tau}$ and the number of clusters in a data set, and that this relationship can be used to find the true number of clusters it contains.

1. Introduction

Cluster analysis is a key tool in data science as it can be used to reveal the class structure of a data set, without requiring labelled samples. Hence, it has been applied to many different problems in areas such as cybersecurity, streaming, Internet-of-Things, anomaly detection, and others (see for instance [3–6], and references therein).

Clustering algorithms can be divided into different categories, such as partitional, hierarchical, and density-based. The latter refers to algorithms following the idea that a cluster is a high-density region of data points (i.e. objects or entities), and that clusters are separated by contiguous regions of low-density. Hierarchical algorithms produce a clustering, usually represented as a tree-like partition of the given data points as well as information regarding the relationship among clusters. Such tree-like relationships can be visualised using a dendogram. Further information regarding hierarchical and density-based algorithms can be found in many sources (see for instance [7–9], and references therein). Here, we focus on partitional clustering, and more specifically, on the k-means algorithm [10,11], which is arguably the most popular partitional clustering algorithm [1,2].

Given a data set X containing n data points $x_i \in \mathbb{R}^m$,k-means produces a partition $S = \{S_1, S_2, \ldots, S_k\}$ of X, such that each cluster $S_l \in S$ is non-empty, and each $x_i \in X$ is included in exactly one subset of S. In any clustering algorithm the general aim is that data points in the same cluster (i.e. a subset of S) should be similar, and data points between clusters should be dissimilar. In the case of k-means, the (dis) similarity between data points is measured using the Euclidean squared distance. That is, given two data points $x_i, x_j \in X$, the distance between them is calculated as follows:

$$d(x_i, x_j) = \sum_{v=1}^{m} (x_{iv} - x_{jv})^2, \tag{1}$$

where x_{iv} and x_{jv} are respectively the values of x_i and x_j corresponding to feature (i.e variable) v, and m is the number of features.

An optimal k-means clustering is that which minimises the value of the following objective function:

$$W = \sum_{l=1}^{k} \sum_{x_i \in S_l} d(x_i, z_l), \tag{2}$$

where z_l is the centroid of cluster $S_l \in S$, computed as the component-wise mean over all $x_i \in S_l$. In order to minimise (2), k-means follows three simple steps:

* Corresponding author.
E-mail addresses: r.amorim@essex.ac.uk (R. Cordeiro de Amorim), makarenkov.vladimir@uqam.ca (V. Makarenkov).

https://doi.org/10.1016/j.neucom.2023.126547
Received 25 November 2022; Received in revised form 3 May 2023; Accepted 8 July 2023
Available online 20 July 2023

1. Select k data points in X at random, and copy their values to the k initial centroids $z_1, z_2, \ldots, z_k$.
2. Assign each $x_i \in X$ to the cluster represented by its nearest centroid.
3. Update each centroid z_l to the component-wise mean over all data points in S_l. If any of the centroids changes its value, go to Step 2.

Let $(\omega_t)_{t=1}^{\tau}$ be a sequence of values such that ω_t holds the value of (2) at iteration t. We have that $\omega_t \geqslant 0$ (i.e. this sequence has a lower bound) and $\omega_{t+1} < w_t$ (i.e. it decreases monotonically). Also, since X is finite, there exists a finite number of different partitions of X. Hence, this sequence (and k-means) not only converges, but it does so in a finite number of iterations. This number of iterations (τ) has, to our knowledge, always been discarded in the data analysis.

The time complexity of k-means is $O(knm\tau)$, where k is the number of clusters, n is the number of data points, m is the number of features, and τ is the number of iterations in the internal loop of the algorithm. Often the maximum number of iterations in the internal loop of k-means is limited by a constant (e.g. $\tau = 100$). However, if the number of iterations τ is not limited, and depends on the speed of the convergence of the objective function (2) to a certain local or global minimum, then the known upper bound on the running time of k-means is $O(n^{km})$ (i.e. it is in general exponential in the number of data points when $km = \Omega(n/\log n)$) [12], whereas the improved lower bound determined by Vattani is $2^{\Omega(n)}$ [13]. These results suggest that in its worst-case scenario k-means requires exponentially many iterations even in the plane. However, in practice, the number of iterations k-means takes to converge is rather linear in the number of data points.

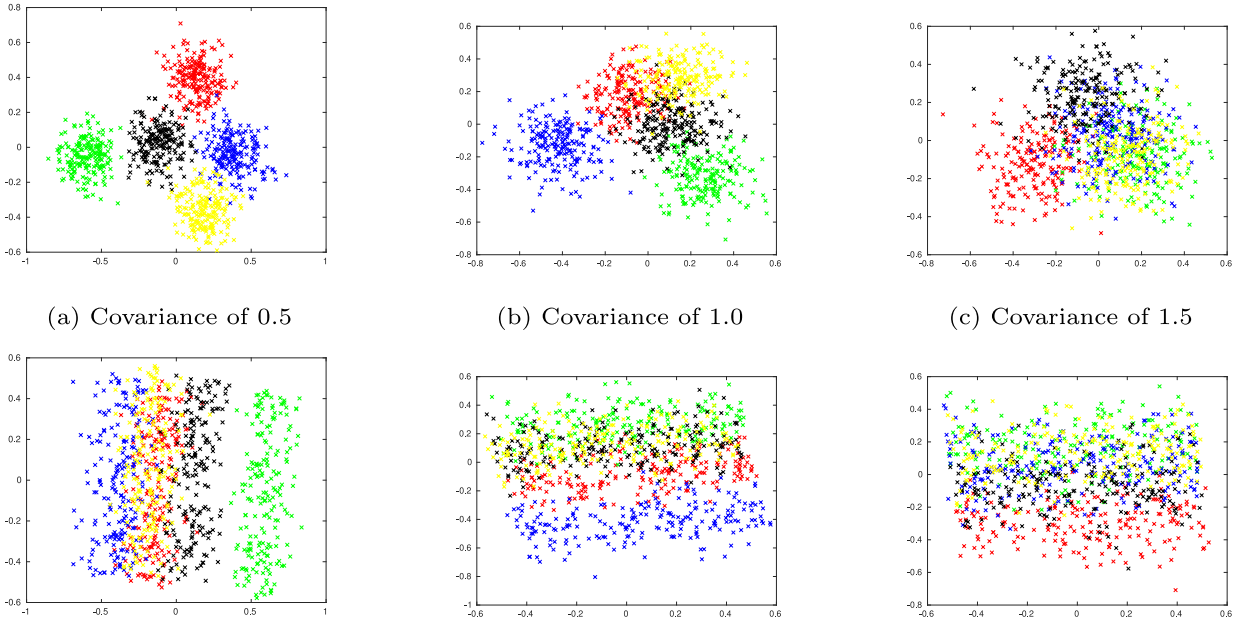
The main contribution of this paper is to show that the number of k-means iterations, τ, is in fact very informative. In our experiments, we use data sets containing Gaussian clusters under different parameter configurations as well as data sets containing uniformly random values. In our four sets of experiments, we show that: (i) with certain data set configurations, τ has a negative correlation with the quality of the recovered clustering (see Section 3); (ii) the lower the covariance within Gaussian clusters, the lower the average value of τ in small data sets, and the opposite holds in larger data sets (see Section 4); (iii) τ can help identify noise features (here, an irrelevant feature composed of uniformly random values) in small data sets containing Gaussian clusters (see Section 5); (iv) there is a relationship between τ and the number of clusters in data sets containing Gaussian clusters (see Section 6).

2. Data sets

In this paper, we perform a number of experiments using k-means (for details see Section 1) and and some other clustering and feature selection algorithms (see Sections 5.1 and 5.2). Generally speaking, we consider three different types of data sets, which we present below.

We began by creating 18 basic data configurations, containing 50 synthetic data sets each. We follow the standard nxm-k. For instance, any data set in the configuration 1000x20-3 contains 1,000 data points, each described over 20 features, with the data points distributed over three clusters. There are three versions of each of these data configurations, leading to a total of $3 \times 18 = 54$. Each version contains spherical Gaussian clusters with diagonal covariance matrices of 0.5,1.0, and 1.5, respectively. A within-cluster covariance of 0.5 leads to tighter clusters, while a within-cluster covariance of 1.5 leads to clusters of higher spread (and by consequence higher overlap), see Fig. 1. In all cases, we generated each centroid component independently from a Gaussian distribution $N(0, 1)$, and each data point was uniformly distributed over the clusters. In total, we created $54 \times 50 = 2,700$ data sets.

For comparison, we also generated data sets under the nxm-k standard containing solely noise features. Here, a noise feature is a feature containing solely uniformly random values. Given these have no within-cluster covariance, there are 18 such configurations with 50 data sets each. The results of our experiments for these data sets appear under "Random data" in Table 2 (more details are given in Section 4). For our experiments with feature selection (see Section 5), we considered the 2,700 data sets containing Gaussian clusters, adding a noise feature to each of them (hence the "+1 NF" in their names). This way, if we request a feature selection algorithm to identify a single irrelevant (noise) feature, we know which one should be selected. Fig. 1 shows some



(a) Covariance of 0.5 (b) Covariance of 1.0 (c) Covariance of 1.5

(d) Covariance of 0.5 with one added irrelevant feature. (e) Covariance of 1.0 with one added irrelevant feature. (f) Covariance of 1.5 with one added irrelevant feature.

Fig. 1. Examples of data sets under the 1000x10-5 configuration (1,000 data points, 10 features, and 5 clusters) with different within-cluster covariances — and their versions with one added noise feature. We plot the data sets over their first and second principal components.

informative examples of the impact of a single noise feature on data structures.

3. Iterations and cluster recovery

In this section, we explore the relationship between the number of iterations *k*-means takes to converge (τ) and the quality of the clusters recovered. Here, we measure clustering quality using the popular Adjusted Rand Index (ARI) [14]. This is a corrected-for-chance version of the Rand Index [15]. An ARI close to zero implies the clustering is (nearly) as poor as a uniformly random partition. Our choice of the evaluation measure is supported by the literature, which shows ARI to be superior to other measures when evaluating clustering quality [16]. In this set of experiments, we carried out *k*-means 100 times on each of the 50 considered data sets and for each of parameter configurations we experimented with. For each *k*-means run, we calculated the value of ARI of the produced clustering, and saved the corresponding number of iterations τ. That is, for each data set we had 100 values of ARI and 100 values of τ. With these, we were able to calculate the correlation index between the values of ARI and τ for each single data set.

Table 1 reports the average correlations over the 50 data sets for each configuration considered, and the related standard deviations. Our experiments show that in some cases there is a considerable negative correlation. For instance, in the case of 100000x100-3 on data sets whose within-cluster covariance is 0.5, we have the most noticeable average correlation of -0.93. One can observe a general pattern that the negative correlation is higher on larger data sets (i.e. those with 10,000 data points or more). In these data sets the lowest negative correlation is -0.55, which is rather meaningful. The correlations in smaller data sets (i.e. those with 1,000 data points) are not always as indicative but we can see a clear pattern. The negative correlation in smaller data sets is higher for the configurations with a higher number of features, a lower number of clusters, and a lower within-cluster covariance. This result follows intuition as such data sets would have a small number of well-formed tight clusters, leading to a higher chance of well-placed initial centroids in *k*-means and by consequence a lower τ. Table 1 also shows that if one adds a single noise feature to a data set, this can have drastic consequences to the correlation we analyse. For instance, 1000x10-5 data sets present a correlation of -0.5 for configuration with clusters whose within-cluster covariance equals 0.5. However, if one adds a single irrelevant feature to such data sets (i.e. configuration 1000x10-5+1NF), then the corresponding correlation drops to 0.07. The same can be observed in other data sets, such as 20000x30-10 and its added noise version 20000x30-10+1NF.

Table 1
Correlation (ρ) between the ARI values (representing clustering quality) and the numbers of *k*-means iterations calculated for each considered parameter configuration. The average values over 50 data sets per configuration are reported.

	Original data sets (no noise)					
	0.5 cov.		1.0 cov.		1.5 cov.	
	ρ	sd	ρ	sd	ρ	sd
1000x5-3	-0.16	0.33	-0.05	0.19	0.07	0.22
1000x5-5	-0.09	0.25	0.12	0.17	0.13	0.14
1000x5-10	0.14	0.12	0.14	0.12	0.09	0.08
1000x10-3	-0.59	0.29	-0.22	0.32	-0.03	0.23
1000x10-5	-0.50	0.18	-0.18	0.24	-0.02	0.16
1000x10-10	-0.14	0.14	0.14	0.12	0.16	0.11
1000x20-3	-0.83	0.12	-0.77	0.18	-0.42	0.35
1000x20-5	-0.73	0.06	-0.61	0.10	-0.42	0.15
1000x20-10	-0.33	0.09	-0.23	0.11	-0.04	0.14
10000x25-3	-0.92	0.04	-0.83	0.30	-0.77	0.32
10000x25-5	-0.85	0.06	-0.84	0.08	-0.80	0.18
10000x25-10	-0.57	0.13	-0.67	0.09	-0.67	0.08
20000x30-3	-0.92	0.05	-0.91	0.15	-0.82	0.31
20000x30-5	-0.85	0.06	-0.87	0.05	-0.88	0.06
20000x30-10	-0.55	0.11	-0.71	0.08	-0.73	0.08
100000x100-3	-0.93	0.04	-0.92	0.04	-0.95	0.04
100000x100-5	-0.83	0.06	-0.87	0.05	-0.87	0.05
100000x100-10	-0.58	0.11	-0.61	0.11	-0.63	0.11
Data sets with an added noise feature						
1000x5-3+1NF	-0.06	0.29	0.02	0.22	0.00	0.20
1000x5-5+1NF	0.08	0.17	0.05	0.13	0.09	0.13
1000x5-10+1NF	0.06	0.12	0.14	0.11	0.11	0.09
1000x10-3+1NF	-0.27	0.21	-0.17	0.23	-0.03	0.24
1000x10-5+1NF	0.07	0.19	0.15	0.16	0.09	0.14
1000x10-10+1NF	0.09	0.09	0.10	0.09	0.13	0.10
1000x20-3+1NF	-0.41	0.21	-0.24	0.18	-0.26	0.24
1000x20-5+1NF	-0.25	0.15	-0.05	0.13	0.01	0.17
1000x20-10+1NF	-0.02	0.12	0.19	0.12	0.16	0.11
10000x25-3+1NF	-0.63	0.20	-0.39	0.28	-0.59	0.20
10000x25-5+1NF	-0.52	0.19	-0.29	0.19	-0.04	0.24
10000x25-10+1NF	-0.22	0.15	0.14	0.21	0.22	0.15
20000x30-3+1NF	-0.75	0.14	-0.37	0.26	-0.58	0.25
20000x30-5+1NF	-0.67	0.17	-0.40	0.15	-0.16	0.25
20000x30-10+1NF	-0.36	0.19	-0.13	0.22	0.22	0.17
100000x100-3+1NF	-0.95	0.02	-0.95	0.02	-0.92	0.05
100000x100-5+1NF	-0.92	0.03	-0.91	0.04	-0.89	0.04
100000x100-10+1NF	-0.59	0.10	-0.62	0.11	-0.64	0.09

Fig. 2 shows the distribution of ARIs with respect to τ (or if the reader prefers, that of τ with respect to ARI) for some of the data sets we experiment with. To generate these results, we ran *k*-means 100 times on each of the 50 data sets for a given configuration. We recorded all values of ARI in relation to the ground truth, and the related values of τ. Each point in Fig. 2 represents a pair (ARI, τ), and there are $100 \times 50 = 5,000$ points on each subfigure. Figs. 2 (a)-(c) present the results for 1000x20-3 with the within-cluster covariances of 0.5, 1.0, and 1.5, respectively. These results show the effect that an increased within-cluster covariance has on the correlation between ARI and τ. These figures also illustrate that the pairs (ARI,τ) form two well-separated clusters, one close to a perfect ARI of 1, and another around the ARI close to 0.45. This supports our view that the high negative correlation we observed (see Table 1) is indeed related to how good the initial centroids of *k*-means are in these small data sets. With good initial centroids the algorithm converges quickly, and because of the nature of the data, this convergence tends to a good clustering solution. If the initial centroids are not good (e.g. all the three initial centroids belong to the same ground truth cluster), then τ is higher and the convergence tends to a poorer clustering solution. Figs. 2 (d)-(f) illustrate the correlation results for the 1000x20-3 + 1NF configuration (i.e. the same data sets with an added noise feature). For these noisy data, the correlation is indeed poorer compared to our previous results. Fig. 2 also shows our results for the 1000x10-5 data configuration for a visual comparison.

Our main conclusion in this section is that there are data sets for which the number of iterations, τ, negatively correlates with ARI. This property concerns particularly two classes of data sets: (i) those that are large (i.e. with at least 10,000 data points); (ii) smaller data sets (i.e. those with 1,000 data points) with a higher number of features, a lower number of clusters, and a lower within-cluster covariance. However, this negative correlation can be lost if the clusters have a higher spread (higher within-cluster covariance) or some noise is added to the data set, even if this takes the form of a single noise feature.

4. Iterations and within-cluster covariance

The *k*-means algorithm is popular, but it is not without weaknesses. One such weakness is that *k*-means will produce a clustering for any data set, even if the data set itself has no cluster structure. This issue raises two questions that we aim to answer in this section: (i) If we were to apply *k*-means to two data sets of equal size (with n objects and m features) - one containing Gaussian clusters and the other composed solely of uniformly random values, would the τ values be approximately the same?; (ii) For data sets containing Gaussian clusters, will the within-cluster variance have an impact on τ?.

Table 2
The average numbers of iterations k-means takes to converge ($\overline{\tau}$) and the related standard deviations (sd). There are 50 data sets for each parameter configuration, and k-means was carried out 100 times per data set (i.e. 100 random starts per data set were performed). Random data columns report the results obtained for data set containing solely uniformly random features.

	Data sets with Gaussian clusters							
	Random data		0.5 covariance		1.0 covariance		1.5 covariance	
	$\overline{\tau}$	sd	$\overline{\tau}$	sd	$\overline{\tau}$	sd	$\overline{\tau}$	sd
1000x5-3	25.06	10.63	9.69	5.84	14.05	7.53	19.35	10.75
1000x5-5	28.38	11.54	16.11	8.82	22.69	10.13	26.07	11.31
1000x5-10	29.06	11.16	23.01	8.97	27.14	9.50	27.28	9.70
1000x10-3	30.10	11.71	6.07	4.44	8.75	4.27	10.32	4.88
1000x10-5	31.97	11.65	9.25	4.94	12.99	6.39	16.89	7.93
1000x10-10	29.92	10.27	13.29	5.76	20.30	8.06	25.36	9.25
1000x20-3	30.16	11.05	5.04	4.97	5.30	4.03	6.37	3.41
1000x20-5	30.67	10.43	7.45	4.96	8.41	4.72	9.53	4.90
1000x20-10	26.74	8.14	8.76	3.64	11.29	5.06	14.51	5.99
10000x25-3	123.58	48.25	9.65	17.14	8.16	13.55	8.06	11.19
10000x25-5	144.85	53.82	18.28	18.60	16.56	17.72	13.42	14.07
10000x25-10	140.22	45.93	24.66	13.01	22.49	13.01	20.80	13.28
20000x30-3	157.96	63.03	12.33	24.30	10.56	20.90	9.94	18.63
20000x30-5	231.11	86.86	24.45	26.48	20.47	25.19	17.80	22.31
20000x30-10	226.41	76.26	35.16	19.30	30.55	19.85	27.65	19.98
100000x100-3	518.21	197.00	24.09	55.17	20.57	54.14	17.56	49.72
100000x100-5	637.53	245.95	48.63	58.61	47.92	61.55	46.07	69.05
100000x100-10	617.13	198.55	76.21	45.08	75.67	47.02	74.95	52.86

It is intuitive to think that the structure of a data set (or the lack of it) has an impact on τ. We first demonstrate that this is indeed the case, and later show how this property can be used in practice (see Sections 5 and 6). To answer both questions raised above, we ran k-means 100 times on each of the 50 data sets we generated for each considered parameter configuration. For each run, we saved the number of iterations k-means took to converge (τ), and calculated their average ($\overline{\tau}$) as well as the standard deviation over these values.

Table 2 reports the results of this set of experiments. Here, we can observe some interesting patterns. For instance, $\overline{\tau}$ is much higher for data sets containing uniformly random values than for data sets containing Gaussian clusters. This suggests that k-means takes, on average, more iterations to converge on data sets that lack structure. The presence of areas of low-density is not a sufficient condition to indicate cluster structure, but such presence is a necessary condition. Hence, a data set containing a cluster structure will have areas of high-density and areas of low-density. With more areas of low-density, we have a higher probability of a centroid moving to (or starting at) an area with less neighbours. Hence, the number of centroid updates is likely to be less than if the data set were to contain solely uniformly random values.

When we analyse the results for data sets containing Gaussian clusters (see Table 2), we can observe another interesting pattern. In small data sets (those with 1,000 data points) the lower the within-cluster variance, the lower the value of $\overline{\tau}$ (and the related standard deviation, in the majority of cases). This seems well-aligned with intuition as the clusters are tighter. However, in larger data sets (those with at least 10,000 data points) the pattern is the opposite. That is, the higher the within-cluster variance, the lower the value of $\overline{\tau}$. It is tempting to think this happens because a within-cluster variance of 1.5 leads k-means to converge quickly but to wrong clusterings in large data sets. This cannot be true as Table 1 shows that large data sets have a higher inverse correlation between $\overline{\tau}$ and cluster quality. Hence, a significant proportion of these low $\overline{\tau}$ convergences are to correct clusterings. With these results, we can now move to a couple of interesting applications.

5. Iterations and feature selection

Feature selection is a major area of research in data science. A feature is one of the m components shared by all data points $x_i \in X$. The general idea behind any feature selection algorithm is to identify whether a given feature is relevant, and to remove it from all $x_i \in X$ should this not be the case. The literature on feature selection is rather vast but tends to focus on supervised methods (see for instance [17,18], and references therein). Here, we are particularly interested in unsupervised feature selection. That is, algorithms capable of assigning a degree of relevance to each feature without relying on labelled samples. In this section, we deal solely with data sets containing Gaussian clusters (to ensure the presence of relevant features). Our two main objectives are to show that there are data set configurations in which: (i) $\overline{\tau}$ can be used on its own to identify a noise feature, producing competitive results; (ii) it is possible to use the information present in $\overline{\tau}$ to improve the results of feature selection algorithms. We tackle these issues by experimenting on different data set configurations to which we added a single noise feature (composed of uniformly random values). We then ask each algorithm considered to identify a single irrelevant feature and calculate the proportion of times it identified the added noise feature as irrelevant.

5.1. Background on feature selection

In this section, we recall the main properties of a few feature selection algorithms, including those we believe to be the most popular. These algorithms allow one to determine how many features should be selected for data analysis.

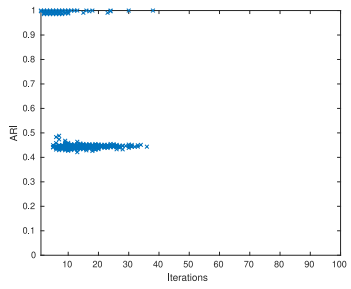
Feature selection using feature similarity (FSFS) [19] is, arguably, the most popular unsupervised feature selection algorithm. It aims at identifying a set of maximally independent features by calculating pairwise feature similarities using the maximum information compression index and applying k-nearest neighbours (k-NN) [20]. Given two features v_1 and v_2, this index is defined as follows:

$$2\lambda_2(v_1, v_2) = \sigma^2_{v_1} + \sigma^2_{v_2} - \sqrt{(\sigma^2_{v_1} + \sigma^2_{v_2})^2 - 4\sigma^2_{v_1}\sigma^2_{v_2}(1 - \rho(v_1, v_2)^2)}, \quad (3)$$

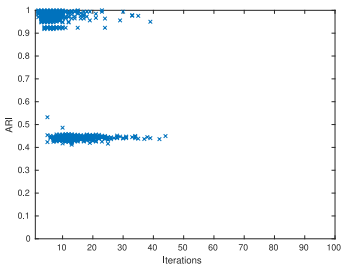
where $\rho(v_1, v_2)$ is the Pearson correlation coefficient between v_1 and v_2, and $\sigma^2_{v_j}$ represents the variance of a feature v_j with $1 \leqslant j \leqslant m$. The value of λ_2 is inversely proportional to the dependency between v_1 and v_2, with the greatest lower bound of zero. Another interesting point, which is rather useful to us, is that this algorithm takes the number of features to be removed as a parameter.

Feature Selection using Feature Similarity (FSFS)

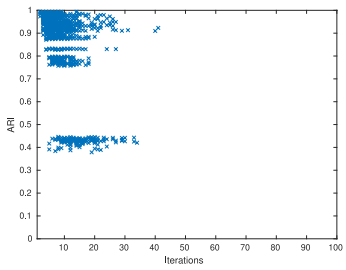
1. Set $V = \{1, 2, \ldots, m\}$, and consider the user-defined value of k (subject to $1 \leqslant k \leqslant m-1$).



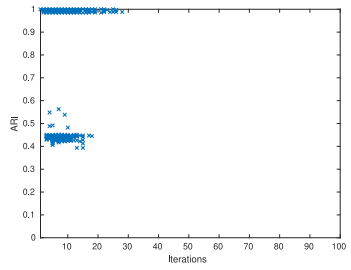
(a) 1000x20-3 data with covariance of 0.5.



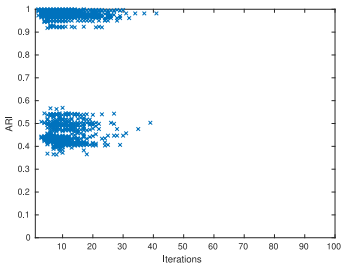
(b) 1000x10-5 data with covariance of 1.0.



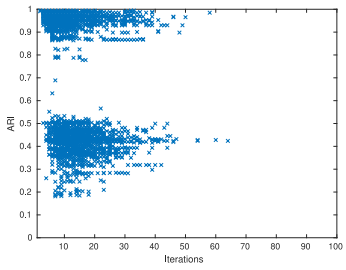
(c) 1000x10-5 data with covariance of 1.5.



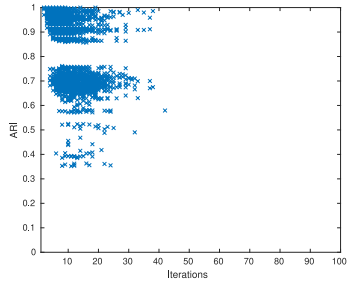
(d) 1000x20-3 data with covariance of 0.5 with one added irrelevant feature.



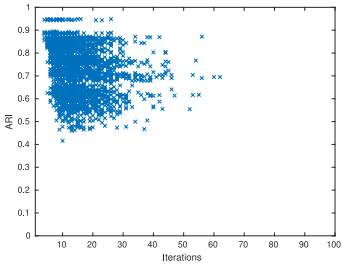
(e) 1000x20-3 data with covariance of 1.0 with one added irrelevant feature.



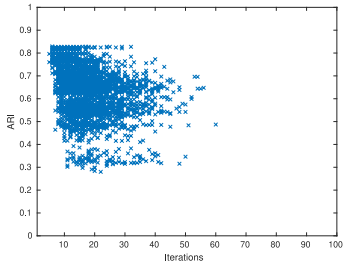
(f) 1000x20-3 data with covariance of 1.5 with one added irrelevant feature.



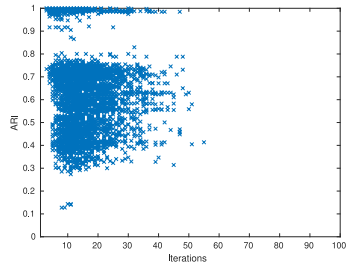
(g) 1000x10-5 data with covariance of 0.5.



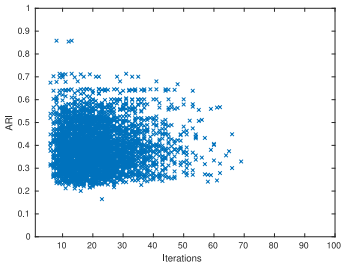
(h) 1000x10-5 data with covariance of 1.0.



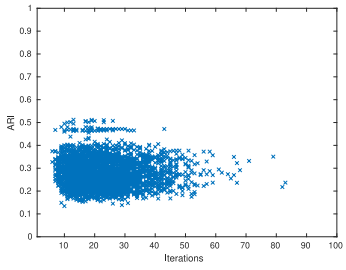
(i) 1000x10-5 data with covariance of 1.5.



(j) 1000x10-5 data with covariance of 0.5 with one added irrelevant feature.



(k) 1000x10-5 data with covariance of 1.0 with one added irrelevant feature.



(l) 1000x10-5 data with covariance of 1.5 with one added irrelevant feature.

Fig. 2. Correlation between ARI and the number of k-means iterations under the 1000x10-5 and 1000x20-3 parameter configurations, with different within-cluster covariances — and their versions with one added irrelevant feature. The data were plotted over their first and second principal components.

2. For each $v \in V$, compute r_v^k. That is, the dissimilarity between v and its k^{th} nearest neighbour in V using (3).
3. Identify the feature v' for which r_v^k is minimum.
4. Remove from V the k-nearest features of v', and set $\epsilon = r_{v'}^k$.
5. Set $k = \min(k, |V| - 1)$.
6. If $k = 1$ go to Step 9.
7. While $r_{v'}^k > \epsilon$
   (a) $k = k - 1$
   (b) $r_{v'}^k = \min_{v \in V} r_v^k$
   (c) If $k = 1$ go to Step 9.
8. Go to Step 2.
9. Return the set of selected features, V.

The intelligent Minkowski weighted k-means (IMWK) [21,3] calculates, independently, the degree of relevance of each feature at each

cluster (w_{lv}). This algorithm applies a weighted version of the Minkowski distance:

$$d_p(x_i, z_l) = \sum_{v=1}^{m} w_{lv}^p |x_{iv} - z_{lv}|^p, \quad (4)$$

and follows the intuitive idea that a given feature may have different degrees of relevance at different clusters. This is modelled using the within-cluster dispersion of each feature given by $D_{lv} = \sum_{x_i \in S_l} |x_{iv} - z_{lv}|^p$, where z_{lv} is the v^{th} component of the centroid of cluster S_l. After calculating all dispersions, the degree of relevance (weight) of each feature at each cluster is given by:

$$w_{lv} = \left(\sum_{u=1}^{m} \left[\frac{D_{lv}}{D_{lu}} \right]^{1/(p-1)} \right)^{-1}. \quad (5)$$

As per the above, the lower the within-cluster dispersion of a feature, the higher its weight [22]. Hence, uniformly distributed features receive a lower weight than those concentrated around their centroids. We can then set the feature with the lowest average weight over all clusters as irrelevant.

Intelligent Minkowski weighted k-means (IMWK)

1. Set $X' \leftarrow X, Z = W = \emptyset, w_{lv} = m^{-1}$ with $l = 1, 2, \ldots, k$ and $v = 1, 2, \ldots, m$, and z to be the Minkowski centre over all $x_i \in X$.
2. Set $z_t = \arg\max_{x_i \in X} d_p(x_i, z), S_t = S_z = \emptyset$.
3. For each $x_i \in X'$, if $d_p(x_i, z_t) < d_p(x_i, z)$, add x_i to S_t. Otherwise, add x_i to S_z. If this step does not change S_t or S_z, then go to Step 5.
4. Update z_t to the Minkowski centre over all $x_i \in S_t$, and each w_{lv} as per (5). Go to Step 3.
5. Add $(z_t, |S_t|)$ to Z and w to W. In any case, set $X' \leftarrow X' \setminus S_t$. If $|X'| > 0$ go to Step 2.
6. Select the k centroids in Z (first component of each pair in Z) whose cluster cardinality (second component of each pair in Z) is the highest. Set $S_l = S_z = \emptyset$.
7. Add each $x_i \in X$ to the cluster S_l whose centroid z_l is the nearest to x_i as per (4). If this produces no changes to any $S_l \in S$, go to Step 9.
8. Update each centroid z_l to the Minkowski centre over all $x_i \in S_l$. Update each w_{lv} following (5). Go to Step 7.
9. Remove the feature v whose average weight over all clusters, that is $k^{-1} \sum_{l=1}^{k} w_{lv}$, is the lowest.

The Minkowski centre of a feature v at a cluster S_l with an exponent p is the value μ that minimises $\sum_{x_i \in S_l} |x_{iv} - \mu|^p$. This algorithm has a parameter, p, used as the Minkowski and weight exponent. In our experiments, we set p equal to two, leading to the squared Euclidean distance. This parameter could be optimised (see [23]) but given our objectives we see no need to do so.

Multi-cluster feature selection (MCFS) [24] is an interesting and popular algorithm that, unlike others, takes into consideration possible correlations between features. It does so by making use of developments in spectral analysis (in particular, manifold learning) and L1-regularised models for subset selection (for details, see [25–28]), preserving the multi-cluster structure of the data set. MCFS requires three parameters, one of which is the number of features to be selected. The other two parameters are the number of eigenfunctions used, and the number of neighbours for a k-NN graph. The original MCFS authors suggest default values of five for both of these parameters but unfortunately this setting produced poor results. With this in mind, we decided to search for the best values for these two parameters (between one and five) for each run of our experiments using the existing labels (the best pair of parameters is that which leads to the best cluster recovery when removing one feature). This clearly biases our MCFS experiments but does not obstruct our objectives.

Multi-cluster feature selection (MCFS)

1. Produce a k-nearest neighbours graph.
2. Solve a generalised eigen-problem and obtain the k top eigenvectors with respect to the smallest eigenvalues.
3. Solve k L1-regularised regression problems, obtaining k sparse coefficient vectors.
4. For each feature $v = \{1, 2, \ldots, m\}$, compute its MCFS score.
5. Return the m' features with the highest MCFS scores, where m' is a user-defined parameter and $m' < m$.

Further details regarding the steps above can be found in the original paper [24]. The above algorithms produce a good baseline for the analysis of our experiments.

5.2. Feature selection using k-means iterations

In order to meet our objectives for this section (stated under Section 5), we introduce two unsupervised feature selection methods. The first of them is solely based on the average number of iterations k-means takes to converge (that is, $\overline{\tau}$). We assume that the data set X has a structure containing Gaussian clusters, which is being concealed by a noise feature. Moreover, we know that the average $\overline{\tau}$ is lower on data sets with a cluster structure than on data sets containing solely noise features (see Section 4). Hence, the feature identified as irrelevant in X is that corresponding to the maximum value of $\overline{\tau}$.

Feature selection via k-means iterations (FSKI)

1. For each $v = 1, 2, \ldots, m$
   (a) Set $X' \leftarrow X$, and remove from each $x_i \in X'$ the feature v.
   (b) Run k-means on X' 100 times, saving the average number of iterations, ($\overline{\tau}_v$), it takes to converge.
2. Return the feature v' corresponding to the lowest value of $\overline{\tau}_{v'}$.

The second algorithm we introduce aims at showing that it is possible to use $\overline{\tau}$ to increase the feature selection capabilities of existing algorithms. In our example, $\overline{\tau}$ is used to re-scale a data set before applying a feature selection algorithm.

Intelligent Minkowski weighted k-means rescaled (IMWKR)

1. For $v = 1, 2, \ldots, m$
   (a) Set $X' \leftarrow X$, and remove from each $x_i \in X'$ the feature v.
   (b) Run k-means on X' 100 times, set $\overline{\tau}_v$ to be the average number of iterations k-means takes to converge.
2. For $v = 1, 2, \ldots, m$, set $r_v = \frac{\sum_{j=1}^{m} \overline{\tau}_j}{\overline{\tau}_v}$.
3. Set $r_v = \frac{r_v}{\sum_{j=1}^{m} r_j}$, that is, normalise each r_v.
4. For each $x_i \in X$ and each $v = 1, 2, \ldots, m$, set $x_{iv} = r_v \times x_{iv}$.
5. Apply IMWK to X.

In the above, Step 2 ensures that if feature v leads to a low average number of k-means iterations ($\overline{\tau}_v$), its r_v will be high. This increases the re-scaled value of v in X (Step 4) and by consequence the dispersion of v. Hence, IMWK (Step 5) is more likely to give a lower weight to v. Thus, the probability of v being chosen as irrelevant increases.

5.3. Experiments and discussion

In this section, we present the results of our experiments applying the algorithms described in Sections 5.1 and 5.2 on data sets containing Gaussian clusters to which we added one noise feature (an irrelevant feature composed of uniformly random values). Table 3 and Fig. 3 report the proportion of times each competing algorithm correctly identified the added noise feature as being the irrelevant one. In our experiments, we ran the non-deterministic algorithms (that is, FSKI and IMWKR) 100 times on each data set. We present only the results for small data sets (i.e. those with 1,000 data points) because the others presented mixed

Table 3
The proportion of times the correct noise feature has been identified by each algorithm (and standard deviation, when appropriate). There are 50 data sets for each of the nine parameter configurations below. A single uniformly random feature, which had to be identified by each algorithm, was added to each data set considered.

	FSFS	MCFS	IMWK	FSKI	IMWKR
	0.5 within-cluster covariance				
1000x5-3+1NF	0.48	0.34	**0.90**	0.88/0.33	**0.90**/0.01
1000x5-5+1NF	0.64	0.52	**1.00**	0.72/0.45	**1.00**/0.00
1000x5-10+1NF	0.86	0.34	0.86	0.58/0.50	**0.92**/0.03
1000x10-3+1NF	0.20	0.68	**1.00**	0.98/0.14	**1.00**/0.00
1000x10-5+1NF	0.22	0.84	**1.00**	0.98/0.14	**1.00**/0.00
1000x10-10+1NF	0.54	0.94	**1.00**	**1.00**/0.00	**1.00**/0.00
1000x20-3+1NF	0.04	0.82	**1.00**	0.20/0.40	**1.00**/0.00
1000x20-5+1NF	0.02	**1.00**	**1.00**	0.22/0.42	**1.00**/0.00
1000x20-10+1NF	0.26	**1.00**	**1.00**	**1.00**/0.00	**1.00**/0.00
	1.0 within-cluster covariance				
1000x5-3+1NF	**0.86**	0.48	0.60	0.62/0.49	0.54/0.03
1000x5-5+1NF	**0.70**	0.36	0.28	0.38/0.49	0.30/0.03
1000x5-10+1NF	**0.92**	0.50	0.34	0.16/0.37	0.26/0.05
1000x10-3+1NF	0.44	0.32	0.88	0.92/0.27	**0.95**/0.01
1000x10-5+1NF	0.68	0.30	0.88	0.94/0.24	**0.98**/0.02
1000x10-10+1NF	0.68	0.24	0.98	0.94/0.24	**1.00**/0.01
1000x20-3+1NF	0.12	0.88	**1.00**	0.86/0.35	**1.00**/0.00
1000x20-5+1NF	0.32	0.96	**1.00**	0.98/0.14	**1.00**/0.00
1000x20-10+1NF	0.46	0.96	**1.00**	**1.00**/0.00	**1.00**/0.00
	1.5 within-cluster covariance				
1000x5-3+1NF	**0.90**	0.50	0.38	0.32/0.47	0.32/0.03
1000x5-5+1NF	**0.94**	0.36	0.10	0.16/0.37	0.10/0.03
1000x5-10+1NF	**0.90**	0.38	0.24	0.26/0.44	0.15/0.03
1000x10-3+1NF	0.62	0.32	0.84	**0.92**/0.27	0.79/0.02
1000x10-5+1NF	0.80	0.26	0.58	**0.88**/0.33	0.72/0.03
1000x10-10+1NF	0.76	0.28	0.78	0.36/0.48	**0.86**/0.04
1000x20-3+1NF	0.36	0.36	**1.00**	**1.00**/0.00	**1.00**/0.00
1000x20-5+1NF	0.56	0.50	**1.00**	**1.00**/0.00	**1.00**/0.00
1000x20-10+1NF	0.58	0.34	**1.00**	**1.00**/0.00	**1.00**/0.00

results without a clear pattern one way or the other.

The first thing we can observe by analysing the results presented in Table 3 and Fig. 3 is that FSKI is rather competitive, on small data sets, when compared to the popular feature selection methods. On the small data sets containing Gaussian clusters with 0.5 within-cluster covariance (i.e. tight clusters), FSKI outperformed FSFS for eight of the nine considered data configurations. On the same data sets, FSKI performed at least as well as MCFS for seven of the nine parameter configurations. This is an impressive performance, particularly given that FSKI is a simple method that uses nothing but the number of iterations k-means takes to converge ($\overline{\tau}$). We can also see that IMWKR (a version of IMWK that uses $\overline{\tau}$ to improve its performance) is the best overall performer, with a noticeably low standard deviation.

Regarding the experiments on small data sets containing Gaussian clusters with a within-cluster covariance of 1.0, the results related to $\overline{\tau}$ (FSKI and IMWKR) are still promising. FSKI performed better than FSFS for six of the nine parameter configurations. This time FSFS performed better on data sets with a low number of features (five). FSKI performed at least as well as MCFS for seven of the nine parameter configurations. Overall, the best algorithm was still IMWKR - this becomes clear after observing the obtained results for data sets with at least 10 features (i.e. the best overall results were obtained for six out of nine configurations). The small data sets with within-cluster covariance of 1.5 contain clusters that are much more likely to overlap. Hence, it is fair to expect a decrease in performance of $\overline{\tau}$-based methods. However, FSKI still outperformed FSFS for five and MCFS for six of the nine parameter configurations. In terms of overall performance, FSKI performed at least as well as all the other methods for five parameter configurations, and

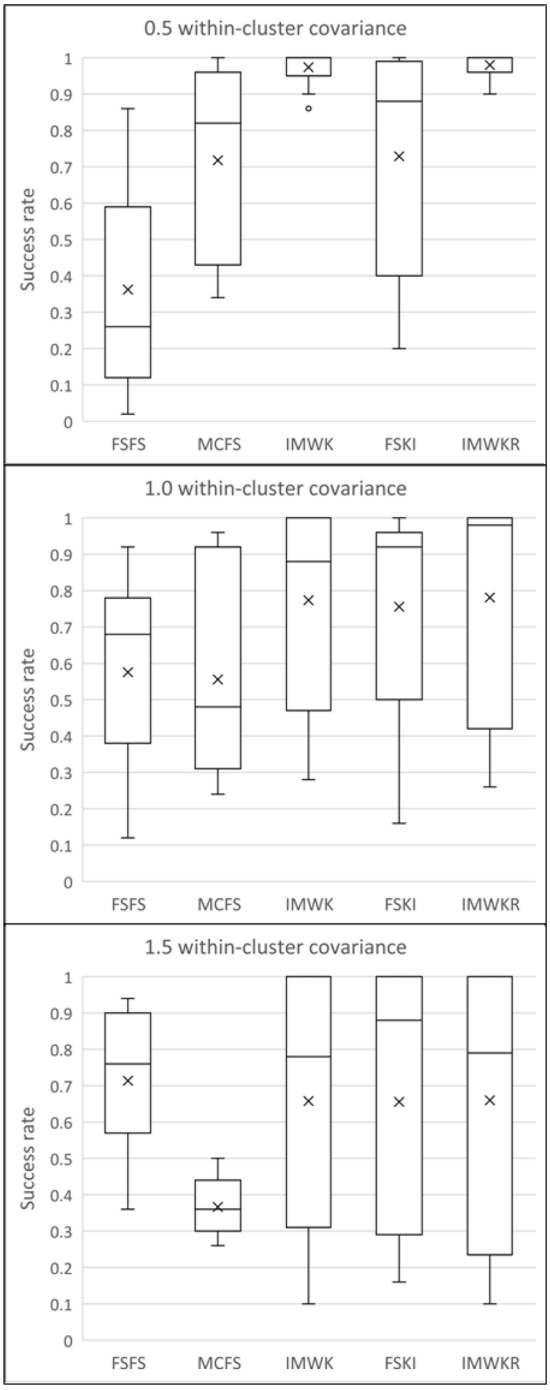


Fig. 3. Boxplot diagrams summarising our results for the proportion of times the correct noise feature was identified by each of the five competing algorithms. A single uniformly random noise feature, which had to be identified by each algorithm, was added to each data set considered.

IMWKR for four of them. Generally speaking, the $\overline{\tau}$-based methods seem to perform particularly well, on small data sets, when the number of features is higher.

6. Iterations and the number of clusters

Identifying the number of clusters in a data set is one of the major problems faced in clustering. Some clustering algorithms, such as k-means require this number to be known by the user beforehand. Others, attempt to identify the number of clusters as part of the clustering

process or as a pre-clustering step (see for instance, [29–32], and references therein). Identifying the number of clusters in a data set is a difficult task, and there is no standard method that works in all cases. This difficulty probably stems from the fact that there is no an agreed definition of what a cluster is, and that a precise definition may depend on the context and the clustering aims (for a full discussion, see [33]). Here, we take the view that clusters are approximately spherical and compact, and can be approximated well by Gaussian distributions. This is not an arbitrary view but rather one based on the algorithm we are analysing here, i.e. *k*-means. In order to identify other types of clusters, one should probably apply a different clustering algorithm.

In this section, we aim at verifying whether there is a relationship between the average number of iterations *k*-means takes to converge ($\bar{\tau}$) and the number of clusters k in X. To meet our objective we ran *k*-means 100 times on each of our data sets (there are 50 data sets for each parameter configuration) with and without supplying *k*-means with the correct number of clusters. In this study, each of our data sets has a correct $k \in \{3,5,10\}$, so we experiment with these numbers. That is, for a parameter configuration such as 1000x5-10 (correct $k = 10$), we carried out *k*-means, supplying it with the following numbers of clusters $k \in \{3,5,10\}$ — in this example $k = 3$ and $k = 5$ are the incorrect numbers of clusters.

Table 4 presents the results of our experiments. An interesting pattern can be observed here. Given any parameter configuration nxm-k (e.g. 1000x5-3, 1000x10-5, etc.), its correct number of clusters is usually that producing the lowest $\bar{\tau}$ in comparison to the data sets with the same n, m, and within-cluster covariance, but different k. For example, the correct k for 1000x10-5 (that is, $k = 5$) has a lower $\bar{\tau}$ than those of the configurations 1000x10-3 and 1000x10-10 (under the same within-cluster covariance). Table 4 shows this is true for 41 out of 54 cases.

Let us analyse the above a bit further. In the case of small data sets (i.e. those with 1,000 data points) the pattern we state holds for 24 out of 27 cases. In the three cases the pattern did not hold, the differences were rather small. For instance, in the column $k = 10$ for data sets with a within-cluster covariance of 1.0 we can see the pattern incorrectly suggests 1000x5-5 has 10 clusters (it has five) given $\bar{\tau} = 26.01$ is the lowest value for rows 1000x5-3, 1000x5-5, and 1000x5-10. However, the correct (see the row for 1000x5-10) has $\bar{\tau} = 27.09$ which is just slightly higher. In the case of larger data sets (i.e. those with 10,000 data points or more), our pattern identified the correct number of clusters in 17 out of 27 cases but this is mostly because of poor performance on data

Table 4
Average number of iterations ($\bar{\tau}$) per value of k supplied to *k*-means. The shaded cells represent cells with the correct k for a given parameter configuration. The experiments in this table were conducted on synthetic data sets containing Gaussian clusters with within-cluster covariance of 0.5, 1.0, and 1.5.

	0.5 within-cluster covariance						1.0 within-cluster covariance					
	$k = 3$		$k = 5$		$k = 10$		$k = 3$		$k = 5$		$k = 10$	
	$\bar{\tau}$	sd	$\bar{\tau}$	sd	$\bar{\tau}$	sd	$\bar{\tau}$	sd	$\bar{\tau}$	sd	$\bar{\tau}$	sd
1000x5-3	9.58	5.80	20.95	9.51	25.00	9.24	13.85	7.36	24.41	10.42	27.02	9.60
1000x5-5	13.74	7.49	15.92	8.62	23.04	8.72	17.51	8.80	22.63	9.84	26.01	9.57
1000x5-10	16.43	7.86	19.00	8.73	22.91	8.96	20.08	9.40	23.73	9.54	27.09	9.74
1000x10-3	6.11	4.20	17.82	6.74	21.43	6.95	8.88	4.39	21.23	8.45	24.60	8.26
1000x10-5	9.60	5.50	9.28	5.05	17.62	5.84	14.64	7.82	12.82	6.40	21.04	7.26
1000x10-10	15.62	7.64	16.18	7.71	13.04	5.82	19.69	9.33	20.86	9.33	20.19	8.09
1000x20-3	4.96	4.87	16.84	5.83	19.05	5.42	5.27	3.90	17.56	6.12	20.04	5.79
1000x20-5	5.37	2.85	7.28	4.87	15.18	4.43	9.14	5.14	8.29	4.76	16.43	5.05
1000x20-10	12.17	6.64	10.53	5.28	8.60	3.67	16.65	7.76	15.91	7.19	11.43	5.03
10000x25-3	9.16	16.55	54.02	21.22	70.74	22.49	7.84	13.21	57.11	21.80	74.14	23.50
10000x25-5	5.81	4.75	17.88	18.54	50.75	16.20	9.75	6.12	16.27	17.39	56.23	17.09
10000x25-10	14.01	8.63	11.91	7.46	24.93	12.79	24.28	13.85	20.77	10.88	22.49	13.20
20000x30-3	13.41	25.37	73.63	30.46	100.81	32.12	8.73	20.19	79.16	32.77	107.19	34.22
20000x30-5	5.55	6.21	24.14	26.21	72.30	24.89	9.00	6.21	20.71	25.38	78.81	25.22
20000x30-10	12.22	7.73	9.99	7.02	35.00	19.27	23.66	14.07	21.18	10.60	31.24	20.01
100000x100-3	23.86	54.65	162.11	74.87	243.67	82.96	21.98	56.38	182.00	76.19	243.89	75.41
100000x100-5	4.25	10.64	46.91	56.66	155.77	58.23	4.75	11.93	51.48	63.52	179.66	63.18
100000x100-10	4.36	1.21	7.25	17.25	70.31	44.25	6.69	2.26	7.70	15.46	82.92	47.51

	1.5 within-cluster covariance					
1000x5-3	19.39	10.88	26.02	10.74	27.53	9.67
1000x5-5	19.49	9.59	25.87	11.20	26.88	9.61
1000x5-10	21.85	10.04	26.32	10.72	27.41	9.65
1000x10-3	10.45	5.26	22.72	8.95	25.71	8.59
1000x10-5	16.59	8.07	16.75	7.80	23.07	7.79
1000x10-10	21.22	8.85	23.27	9.35	25.06	8.87
1000x20-3	6.26	3.16	19.36	7.23	21.50	6.58
1000x20-5	11.81	6.18	9.70	4.99	17.59	5.57
1000x20-10	20.01	9.32	19.30	8.37	14.44	5.78
10000x25-3	8.46	12.10	60.59	22.54	78.86	24.03
10000x25-5	13.56	8.19	13.37	13.98	60.51	19.31
10000x25-10	33.28	16.73	30.51	15.21	20.66	13.48
20000x30-3	10.19	18.98	83.38	32.32	112.61	36.29
20000x30-5	12.27	7.29	17.58	22.49	83.20	26.26
20000x30-10	32.46	18.14	28.99	16.53	26.82	19.40
100000x100-3	17.28	49.33	198.16	79.28	257.62	80.53
100000x100-5	5.14	9.83	47.09	63.15	186.79	58.40
100000x100-10	9.71	5.17	8.14	7.29	80.11	50.65

sets containing clusters with within-cluster variance of 0.5. If we were to ignore these, the pattern would hold true for 14 out of 18 cases. Taking all of the results into account, we can see there is a relationship between $\overline{\tau}$ and the correct number of clusters of X. We find this to be a particularly interesting result because we calculate $\overline{\tau}$ using solely k-means and the given data set X.

Internal cluster validity indices, that is indices requiring nothing external to X and claiming to be related to how good a clustering is (for an extensive review and comparison, see [34]) are often used to determine the number of clusters. However, the way these are applied is rather different. Given a data set X, one usually runs k-means on X with different values of k and then applies one such index to the obtained clusterings in order to determine best one (and by consequence the best value of k). When we want to use the number of k-means iterations $\overline{\tau}$ as a criterion, the approach is different. To find the best number of clusters for X, one should generate data sets as close to X as possible but with different numbers of clusters — the most appropriate number of clusters for X is that which leads to the lowest value of $\overline{\tau}$.

7. Conclusion

K-means remains arguably the most popular clustering algorithm used in scientific and industrial applications [35]. In this paper, we focused on $\overline{\tau}$, that is the average number of iterations k-means takes to converge on a given data set X. This value is always produced when applying k-means, but to the best of our knowledge, it has never been used in the data analysis process. Here, we showed that $\overline{\tau}$ is in fact a very informative parameter of k-means.

First, we discovered that in some cases there is a strong negative correlation between τ and the quality of the recovered clusters, if running k-means multiple times on the same data set. This trend is particularly noticeable in large data sets, or in the case of smaller data sets in those with a high number of features, a low number of clusters, and containing Gaussian clusters with a low within-cluster covariance. However, our experiments also showed that this correlation can be lost if the clusters had a higher spread (higher within-cluster covariance, particularly in small data sets) or if a single noise feature was added to them (see Section 3). We also found that $\overline{\tau}$ is lower on X if the latter contains Gaussian clusters, as opposed to uniformly random values. In fact, we went even further and showed that on small data sets containing Gaussian clusters, the lower the within-cluster covariance (i.e. the tighter the clusters are), the lower $\overline{\tau}$ is (see Section 4). Interestingly, we also showed that the pattern is the opposite for larger data sets. In any case, the structure of X (or the lack of) has an impact on $\overline{\tau}$. Moreover, we also investigated two interesting applications of $\overline{\tau}$. First, $\overline{\tau}$ can be used to help identify an irrelevant feature (e.g. a feature composed of uniformly random values) on small data sets containing Gaussian clusters. We showed this by experimenting with two new methods: (i) removing from X the feature that increases $\overline{\tau}$ the most; (ii) using $\overline{\tau}$ as the base of a feature rescaling procedure in the data pre-processing stage, improving an existing feature selection algorithm — and comparing these two with some popular unsupervised feature selection methods (see Section 5). Second, we showed that there is a close relationship between τ and the number of clusters in data sets containing Gaussian clusters (see Section 6).

We see our work in this paper as potentially leading to improvements in unsupervised feature selection and in the identification of the number of clusters in data sets. In terms of future work, we plan to investigate whether $\overline{\tau}$ could have other practical applications.

CRediT authorship contribution statement

Renato Cordeiro de Amorim: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft. **Vladimir Makarenkov:** Methodology, Formal analysis, Writing - original draft, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] A. Jain, Data clustering: 50 years beyond k-means, Pattern Recognition Letters 31 (8) (2010) 651–666, https://doi.org/10.1016/j.patrec.2009.09.011.
[2] A. Vouros, S. Langdell, M. Croucher, E. Vasilaki, An empirical comparison between stochastic and deterministic centroid initialisation for k-means variations, Machine Learning 110 (8) (2021) 1975–2003.
[3] R.C. de Amorim, C.D.L. Ruiz, Identifying meaningful clusters in malware data, Expert Systems with Applications 177 (2021), 114971.
[4] A. Zubaroğlu, V. Atalay, Data stream clustering: a review, Artificial Intelligence Review 54 (2) (2021) 1201–1236.
[5] Z. Cui, X. Jing, P. Zhao, W. Zhang, J. Chen, A new subspace clustering strategy for ai-based data analysis in iot system, IEEE Internet of Things Journal 8 (16) (2021) 12540–12549.
[6] J. Li, H. Izakian, W. Pedrycz, I. Jamal, Clustering-based anomaly detection in multivariate time series data, Applied Soft Computing 100 (2021), 106919.
[7] P. Bhattacharjee, P. Mitra, A survey of density based clustering algorithms, Frontiers of Computer Science 15 (1) (2021) 1–27.
[8] B. Mirkin, Clustering for data mining: a data recovery approach, Chapman and Hall/CRC, 2012.
[9] F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: an overview, Wiley Interdisciplinary Reviews, Data Mining and Knowledge Discovery 2 (1) (2012) 86–97.
[10] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1, California, USA, 1967, pp. 281–297.
[11] G.H. Ball, D.J. Hall, A clustering technique for summarizing multivariate data, Behavioral Science 12 (2) (1967) 153–155.
[12] D. Arthur, S. Vassilvitskii, How slow is the k-means method?, in: Proceedings of the Twenty-Second Annual Symposium on Computational Geometry, SCG '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 144–153.
[13] A. Vattani, K-means requires exponentially many iterations even in the plane, in: Proceedings of the Twenty-Fifth Annual Symposium on Computational Geometry, SCG '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 324–332.
[14] L. Hubert, P. Arabie, Comparing partitions, Journal of classification 2 (1) (1985) 193–218.
[15] W.M. Rand, Objective criteria for the evaluation of clustering methods, Journal of the American Statistical association 66 (336) (1971) 846–850.
[16] D. Steinley, Properties of the hubert-arable adjusted rand index, Psychological methods 9 (3) (2004) 386.
[17] J. Li, K. Cheng, S. Wang, F. Morstatter, R.P. Trevino, J. Tang, H. Liu, Feature selection: A data perspective, ACM computing surveys (CSUR) 50 (6) (2017) 1–45.
[18] B. Xue, M. Zhang, W.N. Browne, X. Yao, A survey on evolutionary computation approaches to feature selection, IEEE Transactions on Evolutionary Computation 20 (4) (2015) 606–626.
[19] P. Mitra, C. Murthy, S.K. Pal, Unsupervised feature selection using feature similarity, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (3) (2002) 301–312.
[20] N.S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.
[21] R.C. De Amorim, B. Mirkin, Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering, Pattern Recognition 45 (3) (2012) 1061–1075.
[22] R.C. de Amorim, V. Makarenkov, Applying subclustering and lp distance in weighted k-means with distributed centroids, Neurocomputing 173 (2016) 700–707.
[23] R.C. de Amorim, A. Shestakov, B. Mirkin, V. Makarenkov, The minkowski central partition as a pointer to a suitable distance exponent and consensus partitioning, Pattern Recognition 67 (2017) 62–72.
[24] D. Cai, C. Zhang, X. He, Unsupervised feature selection for multi-cluster data, in: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, 2010, pp. 333–342.
[25] T. Hastie, R. Tibshirani, J.H. Friedman, J.H. Friedman, The elements of statistical learning: data mining, inference, and prediction, Vol. 2, Springer, 2009.
[26] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Least angle regression, The Annals of statistics 32 (2) (2004) 407–499.
[27] A. Ng, M. Jordan, Y. Weiss, On spectral clustering: Analysis and an algorithm, Advances in neural information processing systems 14.
[28] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, Advances in neural information processing systems 14.

[29] R. Ünlü, P. Xanthopoulos, Estimating the number of clusters in a dataset via consensus clustering, Expert Systems with Applications 125 (2019) 33–39.
[30] B. Mirkin, Choosing the number of clusters, Wiley Interdisciplinary Reviews, Data Mining and Knowledge Discovery 1 (3) (2011) 252–260.
[31] M.A. Masud, J.Z. Huang, C. Wei, J. Wang, I. Khan, M. Zhong, I-nice: A new approach for identifying the number of clusters and initial cluster centres, Information Sciences 466 (2018) 129–151.
[32] C. Hennig, How many bee species? a case study in determining the number of clusters, in: Data Analysis, Machine Learning and Knowledge Discovery, Springer, 2014, pp. 41–49.
[33] C. Hennig, What are the true clusters? Pattern Recognition Letters 64 (2015) 53–62.
[34] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J.M. Pérez, I. Perona, An extensive comparative study of cluster validity indices, Pattern recognition 46 (1) (2013) 243–256.
[35] P. Berkhin, A Survey of Clustering Data Mining Techniques, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 25–71.

Renato Cordeiro de Amorim is a Senior Lecturer in Computer Science and AI at the University of Essex. He has published a number of papers introducing novel methods following the unsupervised and semi-supervised learning frameworks, with applications in fields such as security, biosignal processing and general data mining. His research is funded by the Royal Society and Innovate UK.

Vladimir Makarenkov is a Full Professor and Director of a graduate Bioinformatics program at the Department of Computer Science at the Université du Québec à Montréal. His research interests are in the fields of Bioinformatics, Operations Research, Artificial Intelligence, and Mathematical Classification.