Applied Intelligence
https://doi.org/10.1007/s10489-018-1238-7

K-means properties on six clustering benchmark datasets

Pasi Fränti[1] · **Sami Sieranoja**[1]

Abstract
This paper has two contributions. First, we introduce a clustering basic benchmark. Second, we study the performance of k-means using this benchmark. Specifically, we measure how the performance depends on four factors: (1) overlap of clusters, (2) number of clusters, (3) dimensionality, and (4) unbalance of cluster sizes. The results show that overlap is critical, and that k-means starts to work effectively when the overlap reaches 4% level.

Keywords Clustering algorithms · Clustering quality · k-means · Benchmark

1 Introduction

The k-means algorithm [1–3] groups N data points into k clusters by minimizing the sum of squared distances between every point and its nearest cluster center (*centroid*). This objective function is called *sum-of-squared errors* (SSE). In this paper, we do not question the suitability of this objective function but merely study how well k-means as an algorithm manages in this task. This approach follows the recommendation in [4] to establish a clear distinction between the *clustering method* (objective function) and the *clustering algorithm* (how it is optimized).

Other algorithms are known to provide better clustering than k-means. However, k-means is very popular for good reasons. First, it is simple to implement, which is an important criterion for choosing the algorithm [60, 61]. Second, people often prefer an algorithm whose limitations are known rather than potentially better algorithms whose limitations are not well known. Thirdly, the local fine-tuning capability of k-means is very effective and for this reason it is also used as a part of better algorithms, such as the genetic algorithm [5, 6], random swap [7, 58] and spectral clustering [42]. Therefore, our results can also be used to better understand those more complex algorithms that rely on the use of k-means.

K-means starts by selecting k random data points as the initial centroids. This initial solution is improved by two steps: assignment and update. In the *assignment step*, every point is put into the cluster whose centroid is closest. In the *update step*, the centroids are re-calculated by taking the mean of all data points assigned in each cluster. Together, these two steps constitute one iteration of k-means. These steps fine-tune both the cluster borders and the centroid locations, see Fig. 1. The steps are iterated a fixed number of times (called iterations), or continued until no further improvement is obtained (convergence).

The time complexity of the assignment step is O(kN) because we need to calculate distance between every input point and centroid. The time complexity of the update step is O(N) to calculate the cumulative sums. Note that we assume the number of dimensions (D) as a constant. In practice, the time complexities should be multiplied by D. The total time complexity is O(gkN), where g is the number of iterations – in our data g =36, on average.

It is well known that the quality of the k-means result depends on the initialization. Bad initialization can cause the iterations of k-means to get stuck in an inferior local minimum. To overcome this problem, numerous initialization strategies have been proposed [33, 35, 37, 39, 55–57]. However, they cannot remove the fundamental limitations of the k-means. Instead of finding better initialization heuristics, much less attention has been paid in the literature under which kind of circumstances k-means fails and when it works.

✉ Pasi Fränti
pasi.franti@uef.fi

Sami Sieranoja
sami.sieranoja@uef.fi

[1] Machine Learning Group, School of Computing, University of Eastern Finland, P.O. Box 111, FIN-80101 Joensuu, Finland

Published online: 26 July 2018

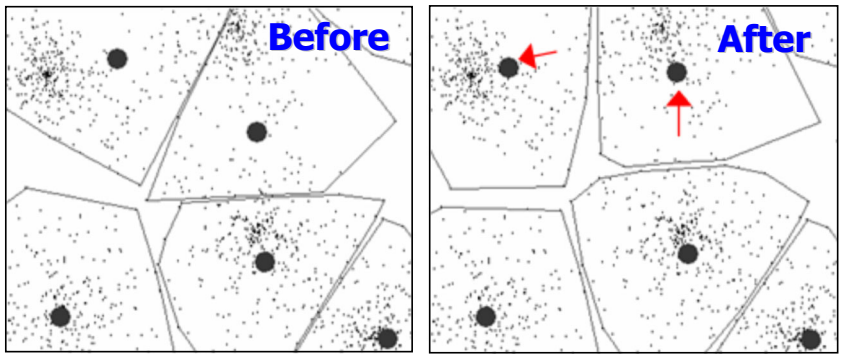


Fig. 1 Example of centroid locations before (left) and after (right) one k-means iteration. K-means moves the centroids towards the actual cluster centers

In this paper, we do no try to fix the problems of k-means. Our goal is to find out how k-means perform under different types of variations in the data. For this purpose, we introduce a clustering basic benchmark. The datasets are chosen so that the SSE objective function can be used for clustering. The sets are challenging enough that most typical heuristics will fail, but easy enough that a good clustering algorithm can solve the correct locations of the cluster centroids.

The benchmark includes the following datasets:

- **A**: varying number of clusters (3 sets)
- **S**: varying overlap (4 sets)
- **G2**: varying overlap and dimensionality (100 sets)
- **DIM**: varying dimensionality with well separated clusters (6 sets)
- **Birch**: varying structure (2 sets)
- **Unbalance**: both dense and sparse clusters (1 set)

Using these datasets, we study the performance of k-means when gradually increasing (1) cluster overlap, (2) the number of clusters, (3) dimensionality, and (4) unbalance in cluster sizes. The *overlap* is an (inverse) estimation of the cluster separation (see (1)); the more separated the clusters are from each other, the lower the overlap. Steinley [8] studied the same four factors along with density, but merely focused on comparing different initialization strategies whereas we study k-means itself.

Users of k-means rarely question its assumed limitations. The first assumption is that the clustering algorithms perform better when the clusters become more separated (less overlap) [8, 46]. This has been proved for the average linkage agglomerative clustering algorithm [47] and for the ratio-cut spectral clustering [48]. The assumption is indeed correct for most clustering algorithms. However, we will show that for k-means, it is the opposite: the greater the separation, the worse k-means performs.

The second assumption is that the number of clusters is k-means' major weakness because it must be given as input [49, 50]. However, the same weakness applies to all algorithms that optimize the same SSE objective function. To solve this problem, one should change the objective function. The k-means *algorithm* itself can be easily applied to multiple values of k; it is just a matter of computation. However, we will show that the success of k-means has an inverse linear dependency on k.

The third assumption is that, dimensionality decreases the performance of k-means [51] or the clustering methods in general [52]. The cause for this is credited to the "*dimensionality curse*", referring either to the sparseness of the data space [53], or that the distances become uniform as dimensions increase [22]. However, concrete studies on the effect of dimensionality on k-means are hard to find. Some good discussions worth reading are found in [9, 35, 54]. We will show that if clusters are well separated (DIM), increasing the dimensions has very little effect. However, if the increasing dimensionality (G2) also decreases the overlap, then the k-means performance degrades. Thus, the mediating factor is the overlap rather than the dimensionality itself.

The fourth assumption is that, k-means produces clusters of relatively uniform size and does not work well with unbalanced cluster sizes [43, 44]. Unbalance is said to be wrongly split and smaller clusters wrongly merged [10, 11]. These papers correctly observe that k-means works poorly with unbalanced data, but they overlook the root cause. The datasets presented in these papers are unsuitable for use with the SSE objective function; even a better algorithm cannot solve these datasets with this function. Therefore, the observation relates to the SSE objective function but has nothing to do with the k-means algorithm itself.

Instead, we show that the k-means *algorithm* indeed works poorly in the case of unbalanced cluster sizes, but for a different reason: the random initialization fails to pick enough centroids from the smaller clusters. Even then, the cause is not the k-means iterations, but the random initialization.

These observations emphasize the need for our proposed benchmark; otherwise researchers will continue using UCI datasets like Wine and Iris, or data that are not suitable for the objective function used. Their results have little

relevance to study clustering algorithms. A controlled benchmark is needed to keep separate the roles of the algorithm and the choice of the objective function.

To sum up, the main research goals of this paper are the following:

- Present basic benchmark clustering datasets.
- Present methodology for measuring success of the clustering.
- Study the behavior of k-means with that methodology.

We focus on the standard k-means. To initialize, we select k random data points as the initial centroids, which is the de facto standard. We briefly consider three other initialization techniques, and explore how much the result can be improved by repeating (re-starting) k-means multiple times from scratch. The purpose is merely to put the k-means performance into context.

The rest of the paper is organized as follows. First, datasets and their properties are presented in Section 2. Methodology on how to evaluate k-means is provided in Section 3. Results are presented and discussed in Section 4. Improvements and alternatives for k-means are briefly considered in Section 5. Conclusions are drawn in Section 6.

2 Data sets and their properties

Luxburg et al. [12] discuss the pros and cons of how to choose benchmarking data. They consider using real world data, classification datasets, and artificial data.

Using real world data makes sense if the clustering is intended to be used in exploratory data analysis, but makes less sense otherwise. Classification datasets can also be misleading because the attributes used in clustering might reveal a completely different clustering structure than that indicated by the class labels. For example, a set of images labeled according to whether an image contains a car or not will provide a completely different result if clustered based on pixel colors or on other low-level features.

The authors in [12] also criticize the use of artificial data for evaluating usefulness of the clustering result. However, we are interested merely in the statistical performance because evaluating usefulness would require some knowledge about the application. Artificial data with known ground truth best serves our purpose because we can then control the parameters we want to study and eliminate the effect of others.

Some of the existing data repositories worth to consider include:

- UCI Machine Learning repository[1]
- Fundamental clustering problem suite [62][2]
- IFCS Cluster Benchmark Data Repository[3]
- Tomas Barton's clustering benchmark[4]
- Marek Gagolewski's clustering benchmark[5]
- Data.World[6]

The UCI machine learning repository contains 76 (May 2018) real world clustering datasets. Most of these are intended for classification and not specifically for clustering. This is problematic because often different structures are revealed by clustering than what the class labels would imply.

The benchmark suite proposed by Ultsch is useful if one wants to demonstrate the problems of the SSE cost function. However, since we have specifically chosen to use SSE, these are not very useful to us. This is because any algorithm minimizing SSE usually fails because the boundary between two clusters is curved whereas the SSE objective function models cluster boundaries as hyperplane.

The other repositories are either application-specific, or collections that merely reproduce the datasets obtained from other sources. However, the main problem is that none of the benchmarks are controlled. Steinley [8] created properly controlled datasets but he did not publish the data.

For this purpose, we present the clustering *basic benchmark*. These datasets have been widely used in studying clustering, but they have not been previously documented in detail. We consider only numerical data and exclude very large datasets to make their use easy. We also do not include any complex datasets because, as the name indicates, it is a *basic* benchmark. All datasets are suitable for the SSE objective function.

We advise the use of this benchmark to test any algorithm claiming to find spherical or Gaussian clusters, to determine whether it can solve these datasets first; the *genetic algorithm* (GA) [6] and *random swap* (RS) [7, 58] can solve them all. The test can also be extended to study how well a certain index can recommend the number of clusters. Although this is already more challenging, the following two simple methods can accomplish it for most of these datasets: WB-index [13] and silhouette coefficient [14].

2.1 Basic benchmark sets

The benchmark datasets are visualized in Fig. 2, and their basic properties summarized in Table 1. All datasets and

[1] https://archive.ics.uci.edu/ml/datasets.html?format=&task=clu

[2] https://www.uni-marburg.de/fb12/arbeitsgruppen/datenbionik/data

[3] https://ifcs.boku.ac.at/repository/

[4] https://github.com/deric/clustering-benchmark

[5] http://www.gagolewski.com/resources/data/clustering/

[6] https://data.world/datasets/clustering

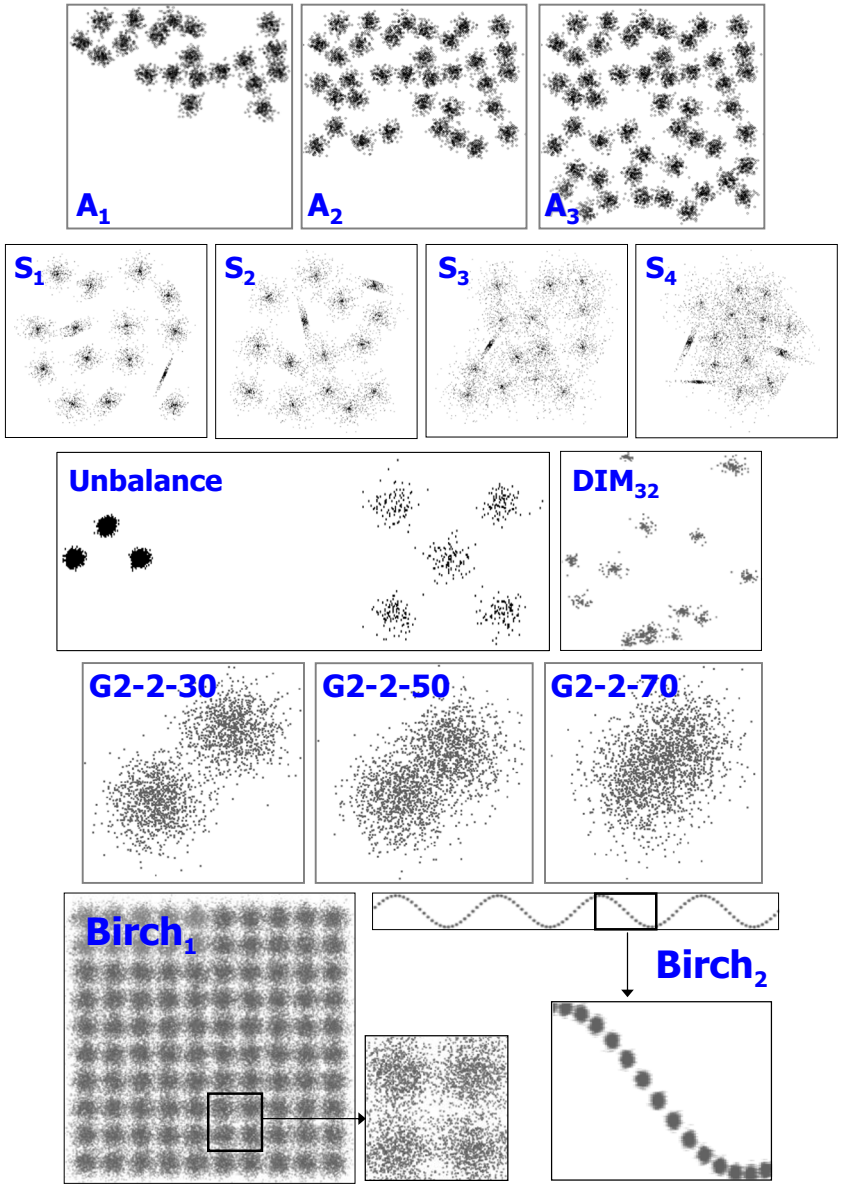


Fig. 2 Illustration of the datasets. All data values are scaled to integers between 0 and a particular maximum value. The plots are symmetric in respect to the scales

their *ground truth* (GT) centroids are publicly available. In the case of G2 sets, the original class labels are also given. For the other sets, the GT partition is obtained by mapping every data point to its nearest ground truth centroid.

We consider ground truth clustering as the one that correctly represents the original parameters used in generating the dataset, i.e. the number of Gaussian distributions and their center points. The datasets were selected so that this

Table 1 Basic clustering benchmark

Dataset	Varying	Size	Clusters	Per cluster	Source
A	Number of clusters	3000–7500	20,30,50	150	[17]
S	Overlap	5000	15	333	[18]
Dim	Dimensions	1024	16	64	[15]
G2	Dimensions + overlap	2048	2	1024	[19]
Birch	Structure	100,000	100	1000	[16]
Unbalance	Balance	6500	8	100–2000	[20]

The data is publicly available here: http://cs.uef.fi/sipu/datasets/

ground truth also matches both the SSE optimal clustering for the dataset and human intuition of the authors. For real world applications there usually is no single correct clustering of data.

A sets: These sets contain spherical clusters with the number of clusters k=20, 35 and 50, so that cluster size (150), deviation (1402), overlap (20%), and dimensionality remain constant. The sets are subsets of each other: A1 ⊂ A2 ⊂ A3.

S sets: These sets contain Gaussian clusters with varying overlap (separation of clusters) from 9% to 44%. Most clusters are spherical, but a few have been truncated to resemble non-spherical Gaussian clusters. The last set (S4) has strong overlap but the clusters are still visible, and solvable by a good algorithm.

G2: These sets contain two Gaussian clusters at fixed locations, each with 1024 points. Overlap was created by increasing the standard deviation of the Gaussian distribution from 10 to 100. The naming practice and the parameters are summarized as follows:

```
Dataset name:G2-dim-sd
Centroid 1:  [500,500, ...]
Centroid 2:  [600,600, ...]
Dimensions:  dim = 2,4,8,16, ... 1024
St.Dev:      sd = 10,20,30, ... 100
```

DIM: These sets contain well-separated clusters in a high-dimensional space with dimensions varying from 32 to 1024. Points within each cluster are random, and sampled from Gaussian distribution. We mainly use DIM032, and the others only when needed. The datasets were first used in [15].

Birch: Three datasets were introduced in [16]. We use the first two, which contain spherical clusters. The third one is more suitable for density-based clustering and has been omitted from our benchmark. The ground truths were not published so we had to estimate them as follows:

Birch1: The clusters form a regular 10x10 grid with the same deviation (21545). Centroids were first optimized using a genetic algorithm [6]. The average distance to the neighbor centroids was calculated as 92247. A grid using this parameter was then manually fit for the data, and the resulting locations recorded as the ground truth centroids. Ground truth partition labels were obtained by mapping every point to its nearest centroid.

Birch2: Centroid locations were first optimized by genetic algorithm, and their average distance in the x-axis was calculated as 9512. The centroids form a sine curve function:

$$y(x) = amplitude \cdot \sin(2 \cdot \pi \cdot frequency \cdot x + phaseshift) + offset$$

It was manually fit using the following parameters:

$$\begin{aligned} offset &= 43659 \\ amplitude &= -37819 \\ phaseshift &= 20.8388 \\ frequency &= 0.000004205 \end{aligned}$$

Ground truth centroids were then plotted in this curve and the corresponding x and $y(x)$ were recorded as the ground truth centroids. Ground truth partitions were obtained by mapping each point to its nearest centroid.

We also created two series of subsets from Birch2. First, we eliminated 1000 random points at a time, resulting in subsets (*b2-random*) with N =1.000 to 100.000. Second, we eliminated one cluster at a time, resulting in subsets (*b2-sub*) with k =1 to 100. These subsets are useful to study the effect of N and k.

Unbalance: The dataset has eight clusters in two well-separated groups. The first three clusters are dense with 2000 points each (st.dev=2043). The five other clusters are sparse with 100 points each (st.dev=6637). The groups are well-separated so that the use of the SSE function results in correct clusters if properly optimized.

2.2 Properties of the datasets

We also calculated the following additional measures to characterize the datasets:

- Overlap
- Contrast
- Intrinsic dimensionality
- H-index
- Distance profiles

Overlap It is possible to count the number of points that are closer to another centroid than its own GT label indicates. This approach is called *misclassification probability* in [21]. This calculation can be done for the G2 datasets (Fig. 3). However, the misclassification rate can be a misleading measure for clustering performance. Often, the data do not have class label. For our data we only have ground truth centroids which result in a 0% misclassification rate. A better solution is therefore needed.

To measure the overlap, we calculated the distance from every point to its centroid (d1) and to its nearest point in another cluster (d2). If this nearest point is closer than its own centroid (d1 > d2), the point is evidence of overlap. Overlap of the dataset is then defined as the number of evidences relative to the total number of points:

$$\text{Overlap} = \frac{1}{N} \sum ov(d_1, d_2) \quad (1)$$

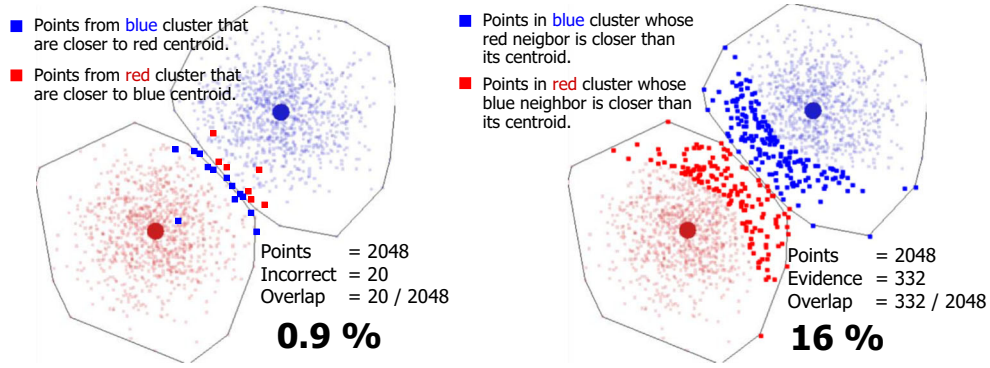


Fig. 3 Overlap for G2-2-30 dataset measured by misclassification probability (left) and by the overlap value (right), resulting in two different values. It is noted that the two values have different orders of magnitude and should not be compared to each other

where $ov(d_1, d_2) = \begin{cases} 1, & d_1 > d_2 \\ 0, & \text{otherwise} \end{cases}$

The overlap values for the benchmark datasets are summarized in Tables 2 and 3.

Contrast This property measures the variation in distances. The contrast of a point is defined as the relative difference in the distances to its nearest ($d_{\min}$) and furthest neighbor ($d_{\max}$). The contrast of the dataset is the average over the entire dataset:

$$\text{Contrast} = median\left(\frac{(d_{\max} - d_{\min})}{d_{\min}}\right) \quad (2)$$

According to [22], the contrast approaches 0 when the dimension increases. Their empirical results suggest that the nearest neighbor can become unstable with as few as 10-20 dimensions. Our observations are quite similar with G2 datasets, see Fig. 4. For all other datasets, the values are reasonably high. Even for the DIM datasets, the minimum (for DIM1024) is 54. These results suggest that the phenomenon does not happen when the clusters are well separated.

Table 2 Properties of the datasets

Dataset	Overlap	Contrast	Intrinsic dim.	H-index
A1	20%	227	1.5	2
A2	20%	261	2.0	3
A3	20%	294	2.5	3
S1	9%	320	2.2	2
S2	22%	257	2.2	3
S3	41%	210	2.0	3
S4	44%	205	2.2	2
Dim32-1024	0%	54 – 167	6.6 – 7.5	7–11
G2	0–66%	0.37 – 494	0.7 – 43.4	2–17
Birch1	52%	799	8.3	3
Birch2	4%	8308	2.6	3
Unbalance	0%	2983	0.4	3

Intrinsic dimensionality Sometimes the true dimensionality of the data is not the same as the number of attributes. For instance, the points in Birch2 are in a two-dimensional space but form a one-dimensional shape along the sine curve. To estimate this true dimensionality, Chavez and Navarro [23] defined the *intrinsic dimensionality* measure as the (squared) average distance between all points divided by the variance of the distances:

$$\text{ID} = \frac{\hat{d}^2}{2\sigma^2} \text{ where } \hat{d} \text{ is the mean and } \sigma^2 \text{ the variance.} \quad (3)$$

The values for Birch2 (2.6) are indeed significantly smaller than that of Birch1 (8.3). The exact value may not match our intuition, but they do characterize the complexity of the data. With the A datasets, the value increases (1.5, 2.0, 2.5) with the number of clusters while remaining almost constant for the S (2.2) and DIM (6.6-7.5) datasets. To summarize, the measure seems to react to the number of clusters and to the complexity of the structure, but not directly to the increase in dimensions.

H-index *The hubness score* of point x has been defined as its indegree (*k-occurences*) in a k-nearest neighbor (k-NN) graph, i.e. the number of points that consider x as its k-nearest neighbor [24]. Contrary to outliers, major hubs usually appear in the central areas. Hubness correlates well with density in low dimensions but not in high dimensions [24]. Distribution of the hubness becomes highly skewed with the increase in dimension [25]. We used 1-NN in our experiments.

Based on the hubness score, we define the *h-index* of the data set similarly as it has been used to measure the scientific impact of scholars and publications [45]. We rank the data points according to their individual hubness score, and choose the h-index as the highest rank for which the score is greater than or equal to its rank. For example, if the hubness scores are 21, 18, 15, 9, **7**, 3, 2, 2... the first value smaller than its rank is four, so the h-index is rank 5.

For our data, the h-index reveals a small difference between high and low dimensional data. The two-dimensional

Table 3 Overlap, contrast, and intrinsic dimensionality values for the G2-datasets (G-dim-sd)

σ\dim	2	4	8	16	32	64	128	256	512	1024
Misclassification rate:										
10	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
20	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
30	1%	0%	0%	0%	0%	0%	0%	0%	0%	0%
40	4%	1%	0%	0%	%	0%	0%	0%	0%	0%
50	8%	2%	0%	0%	0%	0%	0%	0%	0%	0%
60	12%	4%	1%	0%	0%	0%	0%	0%	0%	0%
70	15%	8%	2%	0%	0%	0%	0%	0%	0%	0%
80	19%	9%	4%	1%	0%	0%	0%	0%	0%	0%
90	22%	12%	6%	2%	0%	0%	0%	0%	0%	0%
100	25%	15%	7%	2%	0.1%	0%	0%	0%	0%	0%
Overlap:										
10	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
20	3%	0%	0%	0%	0%	0%	0%	0%	0%	0%
30	16%	5%	0%	0%	0%	0%	0%	0%	0%	0%
40	26%	19%	3%	0%	0%	0%	0%	0%	0%	0%
50	32%	31%	13%	1%	0%	0%	0%	0%	0%	0%
60	39%	43%	27%	4%	0%	0%	0%	0%	0%	0%
70	43%	50%	40%	11%	0.1 %	0%	0%	0%	0%	0%
80	46%	58%	50%	20%	0.8 %	0%	0%	0%	0%	0%
90	47%	61%	60%	31%	2.0 %	0%	0%	0%	0%	0%
100	47%	66%	64%	39%	4.4 %	0%	0%	0%	0%	0%
Contrast:										
10	172	53.8	22.3	14.0	10.7	8.9	7.9	7.3	6.9	6.7
20	161	31.6	12.4	7.4	5.2	4.3	3.7	3.4	3.1	3.0
30	131	22.8	9.1	5.2	3.5	2.7	2.3	2.1	1.9	1.8
40	123	20.0	7.5	4.0	2.7	2.1	1.7	1.5	1.3	1.2
50	115	18.0	6.3	3.4	2.2	1.7	1.3	1.1	1.0	0.9
60	108	16.2	5.5	3.0	1.9	1.4	1.1	0.9	0.8	0.7
70	123	16.4	5.2	2.7	1.8	1.2	1.0	0.8	0.7	0.6
80	122	16.0	5.2	2.6	1.6	1.1	0.9	0.7	0.6	0.5
90	116	15.0	4.9	2.5	1.5	1.0	0.8	0.6	0.5	0.4
100	110	15.3	4.8	2.4	1.4	1.0	0.7	0.6	0.4	0.4
Intrinsic dimensionality:										
10	0.8	0.8	0.8	0.9	0.9	0.9	0.9	0.9	0.9	0.9
20	1.1	1.3	1.4	1.5	1.5	1.5	1.5	1.5	1.5	1.5
30	1.4	1.9	2.2	2.4	2.5	2.5	2.6	2.6	2.6	2.6
40	1.6	2.4	3.1	3.6	3.9	4.1	4.2	4.2	4.2	4.3
50	1.7	2.8	4.1	5.0	5.9	6.3	6.6	6.7	6.9	6.8
60	1.8	3.1	5.0	6.6	8.1	9.2	10.0	10.3	10.5	10.5
70	1.8	3.3	5.6	8.5	11.0	13.1	14.4	15.1	15.7	15.9
80	1.8	3.5	6.1	9.9	13.9	17.3	19.6	21.4	22.1	22.3
90	1.8	3.6	6.4	11.2	16.6	22.3	26.9	29.1	30.5	31.5
100	1.8	3.6	6.7	12.2	19.1	26.9	34.0	39.1	41.2	43.4

Table 3 (continued)

σ\dim	2	4	8	16	32	64	128	256	512	1024
H-index:										
10	4	3	4	5	9	11	14	16	15	14
20	3	3	4	6	9	12	14	15	15	16
30	2	3	3	6	10	11	14	15	14	15
40	3	3	4	6	10	11	15	13	17	16
50	2	3	4	7	9	12	13	14	15	17
60	2	3	4	6	9	13	14	14	16	16
70	2	3	4	6	10	11	12	13	14	16
80	2	3	4	6	9	12	15	14	16	17
90	2	3	4	6	9	12	15	14	18	14
100	2	3	4	5	9	11	15	15	15	15

The first parameter (rows) represents the standard deviation (σ) of the Gaussian distribution and the second parameter the number of attributes (dimensions)

data simply do not have hubs. The DIM datasets have slightly higher values and shows an increasing trend with the dimensionality (7, 9, 9, 10, 10, 11). The G2 datasets also have this logarithmic increase with the dimensions.

Distance profiles Steinbach et al. [9] used histograms of the distances to estimate whether the data have clusters. Data that contain clusters tend to have two peaks: one representing the distances inside the clusters (local distances), and another representing distances across different clusters (global distances).

Histograms of the A datasets indeed have a tiny peak for the local distances, but it is hardly noticeable as it is overwhelmed by the global distances, see Fig. 5. The same peak is also noticeable in S1 and S2 but not in S3 and S4 due to the overlap. The unbalance dataset has three peaks: local distances in the dense clusters, local distances in the sparse clusters, and the flat area for the global distances.

With high dimensional datasets, the peaks are clearly visible. With G2 and D=128, the peaks remain separable even with the maximum overlap (sd=100). Due to the strong overlap, a single attribute has a high probability of being closer to the neighbor cluster, but for the entire point to become misclassified, roughly $> 50\%$ of its attributes should be closer to the wrong cluster. Assume that the probability for a single attribute to be on the wrong side is $p = 25\%$. The probability for the same to happen for more than half of the attributes is already less than 10% already when D=8. Table 3 displays this phenomenon clearly with the G2 datasets.

In general, clusters with varying densities and distances from each other will produce multiple peaks, and overlap causes the peaks to merge. Therefore, the distance profiles can be useful in revealing when the data has well-separated clusters.

Summary Tables 2 and 3 collect the statistics on the datasets. Increasing the number of clusters also increases the intrinsic dimensionality and contrast (A sets). Increasing the overlap reduces contrast but does not have a clear effect on the other measures. Dimensionality significantly reduces contrast when the clusters overlap, but reduction is decreased if the clusters are separated. Dimensionality also reduces overlap, but does not seem to have a direct influence on the intrinsic dimensionality.

3 Evaluation of clustering results

Clustering results are usually evaluated by the SSE objective function, or, if ground truth partition is known, by an external validity measure like the *adjusted Rand index* (ARI). However, these indexes do not necessarily reveal how significant the error is. Minimizing SSE is an NP hard problem both in D-dimensions [26] and also in the two-dimensional special case [27]. However, finding the exact optimum is usually not needed.

A recent invention, the *centroid index* (CI), is better suited for this purpose [28]. It has a clear binary output for every centroid, indicating whether the centroid is correctly located in respect to the ground truth. Based on CI, we define the *success rate* (%) as the relative number of times the algorithm reaches CI=0. In literature, it is common to use 2-d visual examples but leave the analysis to the reader. The success rate formalizes this analysis and extends it naturally to higher dimensions.

We use the following notations:

$x_i =$ i^{th} data point

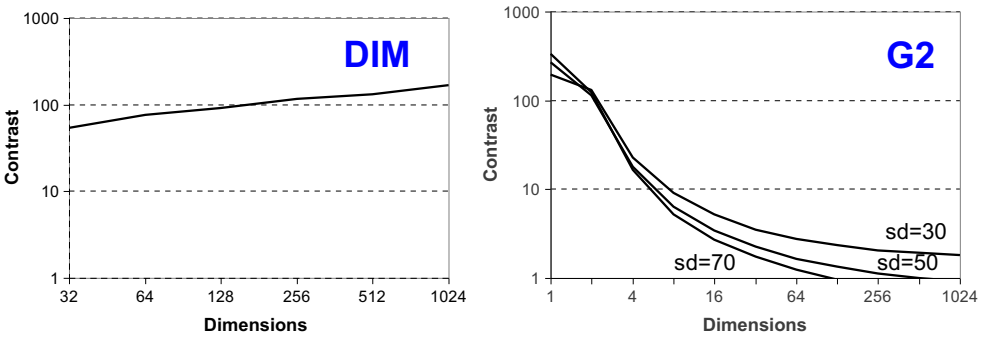


Fig. 4 Contrast values of G2 datasets decreases rapidly with dimensions (right). The values of the DIM datasets do not have a similar behavior (left). The more the overlap, the stronger the effect

$c_j =$ j^{th} centroid
$N =$ number of objects
$D =$ number of dimensions
$k =$ number of clusters

We use the following measures:

Internal measures:

- Sum of squared distances (SSE)
- Normalized mean square error (nMSE)
- Approximation ratio (ε)

External measures:

- Success rate (%)

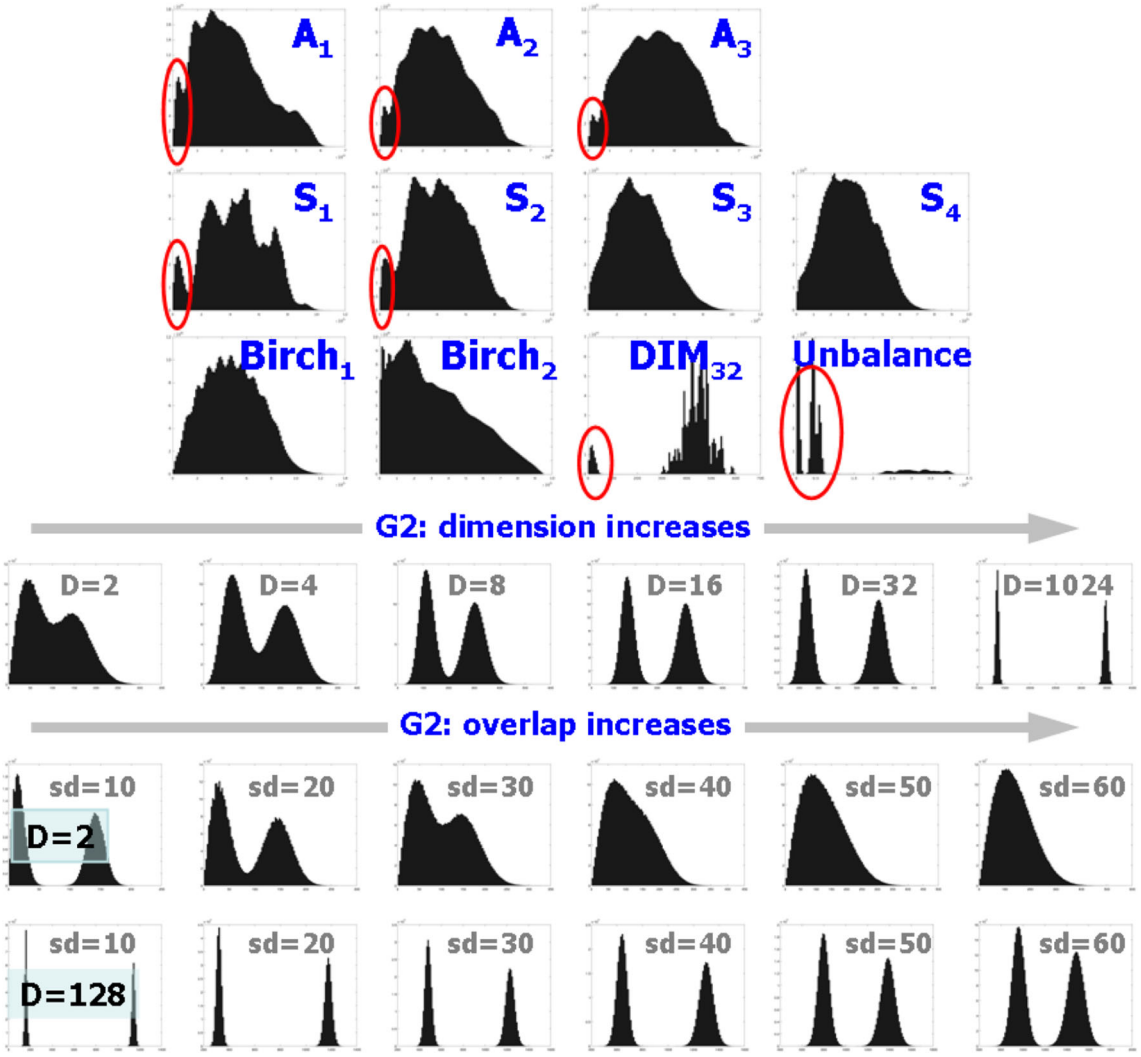


Fig. 5 Distance profiles of the datasets

- Centroid index (CI)
- Adjusted Rand index (ARI)

3.1 Internal measures

Internal measures depend only on the data points. These measures are all variants of the same basic objective function that the clustering aims to minimize:

$$\text{SSE} = \sum_{i=1}^{N} ||x_i - c_j||^2 \quad (4)$$

where xi is the data point and cj is its nearest centroid. In several of our previous works [6, 7, 15, 17, 18], the results have been reported using the normalized version of the mean squared error (nMSE):

$$\text{nMSE} = \frac{SSE}{N \cdot D} \quad (5)$$

In our previous papers, the nMSE has been referred to by the name MSE, because the methods were originally developed for image vector quantization where it was natural to normalize the error per pixel (N vectors in image, D pixels in vector). Presenting nMSE allows for direct comparison to the results in those papers, especially in [18].

Using the objective function values as such might not tell much about the result. For instance, if one algorithm gives SSE=3.39 and another SSE=2.40, then we can say the latter works better. But how much better? Is the difference significant? To answer these questions, we also measure the SSE relative to the optimal clustering (SSE_{opt}):

$$\varepsilon\text{-ratio} = \frac{(SSE - SSE_{opt})}{SSE_{opt}} \quad (6)$$

The motivation is to compare the results with the theoretical results obtained for approximation algorithms. For instance, Kanungo et al. [30] proposed a local search based algorithm which achieves a factor of $(9+\varepsilon)$, which was claimed to be the best-known approximation algorithm for SSE. For clustering three-dimensional sequences, even better polynomial were presented both 4-approximation and $(1+\varepsilon)$-approximations in [31]. Awasthi [32] showed that there exists a constant $\varepsilon > 0$ such that it is NP-hard to approximate the objective function within a factor of $(1+\varepsilon)$. Log(k)-approximation was given in [33]. Even with two clusters (k =2) it already gives ε =1.0, and with S datasets ε =3.9.

Although in our work the ε-ratio is not a proven upper bound, it gives the results in scale that better demonstrate how high the error is. To calculate SSE_{opt}, we could use the ground truth centroids. However, their locations may be different than the optimal locations that minimize SSE. For this reason, we use the result of genetic algorithm (GA) [6] to estimate the SSE_{opt}.

3.2 External measures

We use the *centroid index* (CI) as our primary measure of success. It counts how many real clusters are missing a centroid, or alternatively, how many clusters have too many centroids. The CI-value is the higher of these two numbers [28], see Fig. 6. This provides a much clearer intuition about the result. Specifically, if CI=0, we conclude that the result is correct clustering. We say that the algorithm then solves the problem. Sometimes we normalize CI by the number of clusters, and report relative CI = CI/k.

The success rate is then defined as the relative number of times (%) an algorithm reaches the correct clustering (CI=0). It measures how often the best possible clustering result is found. Since perfect results are always not necessary, we use CI as the primary measure, but success rates can be useful when comparing performance of different algorithms.

Since CI measures only cluster level differences, we also use the adjusted Rand index (ARI) to give more detailed point level differences. Matlab implementation of ARI can be found in [29].

4 K-means properties

We next study empirically how much the performance of k-means depend on the following factors:

- Overlap of clusters
- Number of clusters
- Dimensionality
- Unbalance of cluster sizes

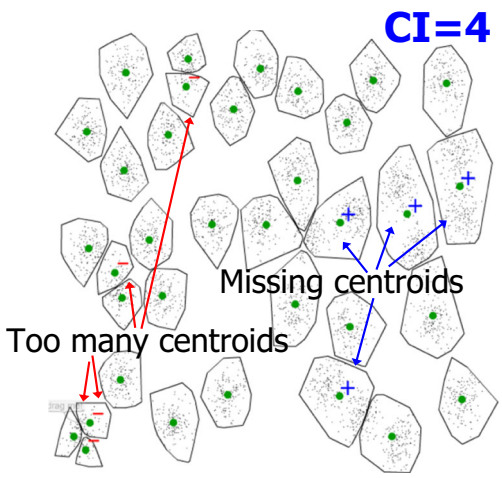


Fig. 6 Example of a typical k-means result for the A2 dataset. The corresponding measures for this are: CI = 4, SSE = 3.08, nMSE=2.93, ε = 0.52

Table 4 Summary of the k-means results (averaged over 5000 runs)

Dataset	Success	Clustering quality			Objective function		
		CI	rel-CI	ARI	SSE	nMSE	ε
A1	1%	2.5	12%	0.82	1.98	3.31	0.64
A2	0%	4.6	13%	0.82	3.39	3.23	0.67
A3	0%	6.6	13%	0.82	4.90	3.27	0.69
S1	3%	1.9	13%	0.85	18.84	18.84	1.11
S2	11%	1.4	9%	0.86	19.79	19.79	0.48
S3	12%	1.3	9%	0.84	19.51	19.51	0.14
S4	26%	0.9	6%	0.84	17.00	17.00	0.07
Unbalance	0%	3.9	49%	0.64	2.10	1.61	8.81
Birch1	0%	6.7	7%	0.85	10.95	5.47	0.18
Birch2	0%	16.6	17%	0.81	15.75	7.87	2.45
Dim32	0%	3.6	22%	0.76	16.5	504	68.34
Average:	5%	4.5	15%	0.81	—	—	7.60

The overall results are summarized in Table 4. We see that k-means rarely solves the problem. The best case is S4, where k-means manages to solve the problem 26% of the times. Sets A2, A3, Unbalance, Birch1, Birch2 and DIM32 were not solved even once. On average, 4.5 centroids are incorrectly located. We next study the results in more detail.

4.1 Overlap

On average, the k-means algorithm finds the correct solution (CI=0) about 5% of times. The easiest are the S sets. The more overlap, the more often it finds the solution (26% of times with S4). The datasets with high separation (Dim32, Unbalance, Birch2) are problematic: k-means never finds the correct clustering and the CI-values (3.6, 3.9, 16.6) are quite high. The reason why overlap is good is demonstrated in Fig. 7.

Further tests with G2 datasets show a clear dependency between overlap and success. Since these datasets have only two clusters, there is no sense in applying clustering with $k = 2$ (too trivial to solve). Instead, we allocated $k = 4$ centroids with a 3:1 balance so that the first group had three centroids and the second group had one. We then ran k-means and checked whether it found the better (in minimizing SSE) 2:2 balance by moving one of the three centroids to the second group. The results in Fig. 10 show that a higher overlap implies better performance.

Too much overlap would eventually make the clusters merge and any clustering algorithm (not only k-means) would fail. With G2 data, this merging has not yet happened (highest observed overlap is 66%). In some cases, however, the clusters are barely recognizable when std=100, and other, better algorithms than k-means, also start to lose their accuracy.

We further plot all results of the G2 datasets with $D = 2$ to 32 dimensions and sd=10-100 in Fig. 8 according to the measured overlap of the data (x-axis) and the success rate (y-axis). There is strong correlation (0.91) between overlap and success. The success of k-means reaches 1% at the point where the overlap exceeds 4%. This appears to be approximately the critical point for the success of k-means with this data.

In brief, overlap is very important factor to predict the performance of k-means. However, it is not the only factor affecting the performance. Even with relatively high overlap (20%), k-means still failed with the A sets.

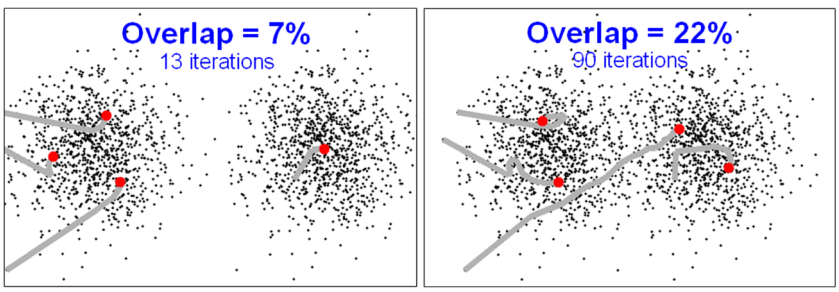


Fig. 7 Illustration of the effect of overlap for k-means. The gray trajectories show the movement of the centroids during the iterations. In both cases, only one initial centroid is on the rightmost cluster and only when there is sufficient overlap, one additional centroid can move across the clusters

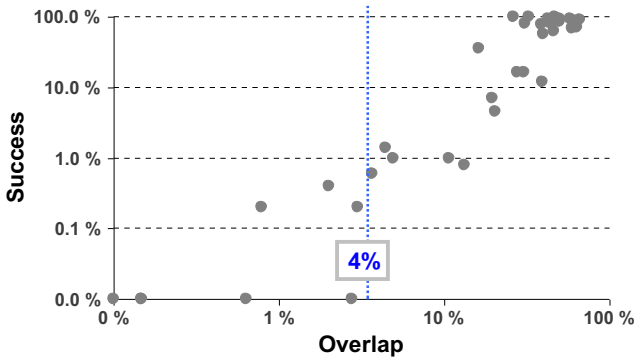


Fig. 8 Dependency of overlap and success rate on G2 datasets. The location (4%) where the success rate reaches 1% is marked

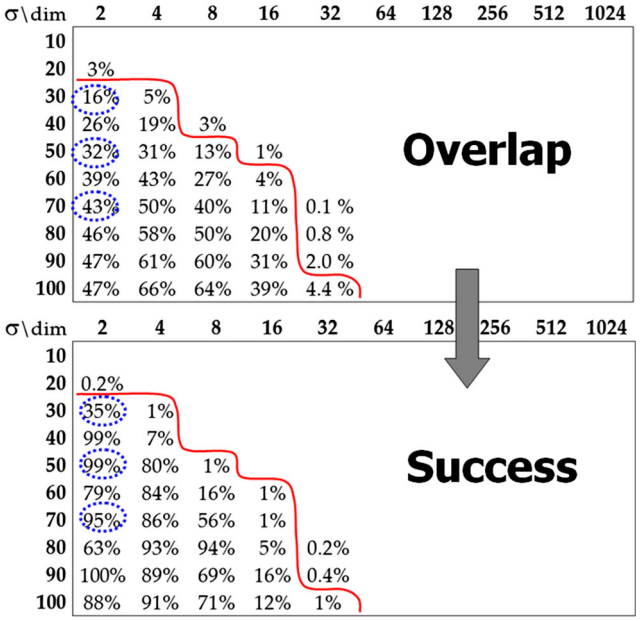


Fig. 10 Success rates for all G2 datasets (bottom chart). Rows are the standard deviation (σ) of the Gaussian distribution and columns the dimensionality. The overlap values (in %) are also shown to demonstrate the strong influence of overlap on success. The blue circles refer to the datasets shown in Fig. 2

4.2 Number of clusters

K-means can occasionally solve the S sets but almost never solve the A sets even if their overlap level is not significantly different. The reason is the higher number of clusters. The results of SSE and ARI reveal little. However, the CI-values and the success rate reveal that there is a clear linear dependency on the number of clusters. This can be seen with A1-A3 datasets: the CI-values 2.5, 4.5, 6.6 all correspond to the relative CI-value of 13% in all three cases.

Figure 9 demonstrates how the CI-value depends on the size of the data (left), and on the number of clusters (right). The value has almost no dependency on the size of the data, but it increases linearly with the number of clusters. The increase is steady and the relative CI-value (CI/k) is roughly 16.6%. This means that one cluster out of six is missed by k-means, on average. The same trend happens also with repeated k-means, for which the relative CI-value converges to 10.9%.

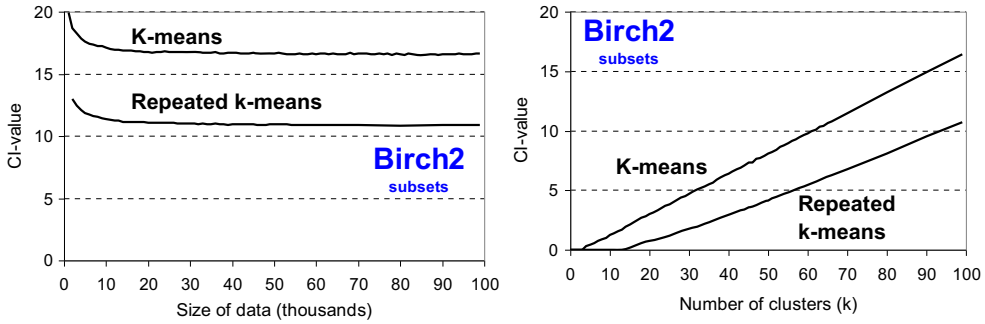
4.3 Dimensions

The DIM sets confirm the same observation that overlap is necessary for k-means. However, the dimensionality does not affect the results, and the CI-value is almost constant; when dimensionality increases from D =32 to 1024 the success rate remains at 0% and the CI-value has only minor variations: 3.6, 3.5, 3.8, 3.8, 3.9, 3.7. It seems that the overlap matters more than the dimensionality itself.

For all G2 sets, k-means is successful with low dimensional datasets when the clusters overlap, see Fig. 10. For instance, G2-4-50 is solved 8% of the time whereas G2-16-50 is never solved. However, the explaining factor is again the overlap (see Table 3), which is reduced from 31% to 1%, below the critical point. Similar observation was made in [22] by concluding that the lack of contrast is a major problem with high dimensional data for many applications, and it start to happen already for 10-15 dimensional data.

To sum up, a higher dimensionality weakens k-means but the reason is the lack of overlap. For example, DIM sets have no overlap so the k-means never succeeded.

Fig. 9 Dependency of CI-values on the size of data (B2-random), and on the number of clusters (B2-sub)



4.4 Unbalance

The unbalance dataset demonstrates another weakness of k-means. The problem is not the density itself but the unbalance of frequencies, together with the separation of the clusters. If no centroids are selected from the small clusters, k-means can move only one centroid into this area – the four others that are needed would get stuck in the dense area, see Fig. 11.

This behavior can be explained by the probabilities and the problem is rooted in the random initialization. The probability that a randomly selected point comes from the low frequency area is only 500/6500 = 7%. The probability to select all five is diminishingly low. Even if this selection occurred, the centroids might still be incorrectly located. In our experiments, k-means solves the unbalance dataset only six times out of 100,000 trials (0.006%).

5 Improving k-means

There are three obvious ways to improve k-means:

- Better initialization
- Repeated k-means (RKM)
- Replace k-means by another better algorithm

Although studying different initialization strategies is out of the scope of this work, we briefly consider alternative initialization strategies, and to show to what extent the problems of k-means could be fixed by the above-mentioned options.

In our view, initialization should be trivial, or at least very easy to implement, and free from any parameters (besides k). For instance, there are algorithms that start from the center of the data as the first centroids [34], and then consider other points in a certain order for the next candidate. A new centroid is created if the point is farther than a given distance to previous centroids. Other similar sorting heuristics also exist, but we did not consider them because of the need to set a distance threshold.

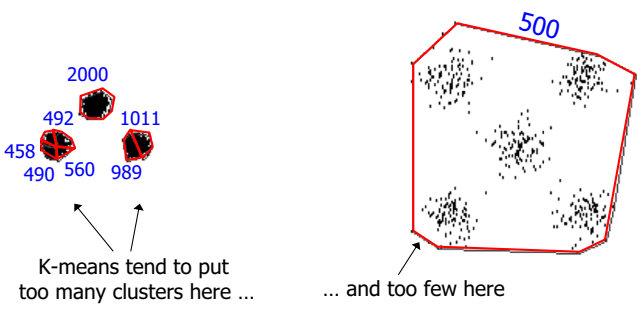


Fig. 11 Unbalanced cluster sizes are a problem for k-means

It is also possible to repeat k-means multiple times. The idea is simply to run k-means several times, each time with a different initial solution, and then keep the best result. This approach requires the initialization technique to include randomness to obtain different solutions. Many researchers consider the repeats as an obvious and necessary improvement to the k-means, where better clustering is obtained at the cost of increased processing time. The approach is often credited to [35], although it has likely been used by many others earlier including [59]. We call this variant repeated k-means (RKM). In all of our tests conducted here, we applied 100 repeats.

5.1 Initialization techniques

We consider the following initialization techniques:

- Random centroids (Ran) [1, 2]
- Steinley's algorithm (Ste) [36]
- Further point heuristic (Max) [37]

Random centroids [1, 2] is the standard initialization technique that we have used in the earlier part of this paper. An alternative variant is random partitions [3], which puts every point into a randomly chosen cluster, and then calculates the centroids of these partitions. Steinley's variant [36] repeats this process 5000 times and selects the result with smallest SSE. In our implementation, we use the random number generator in [38].

A third popular technique is the furthest point heuristic. It selects the mean of all points as the first centroid. Then, at each step, the next centroid is selected as the furthest point (max) from its nearest (min) centroid. This is known as *Maxmin* [39]. We use a variant in which the first point is chosen randomly. This adds randomness to the process, which is useful if we want to repeat k-means multiple times.

5.2 Results

The results of different initializations are summarized in Table 5. In general, Steinley's method works poorly. It has a much better success rate on the S2 and S4 sets but when the number of clusters increases, it creates many empty clusters, which results in high CI-values. The worst case is Birch2, for which 76% of all centroids are incorrectly located.

Maxmin is the only heuristic that solves one set with 100% certainty (DIM32). It works significantly better than the random heuristic but still makes 2.2 errors in centroid allocations, on average. It also works poorly when the number of clusters increases (A2, A3, B1, B2). It works significantly better with Birch2 (4% overlap) than with Birch1 (52% overlap) due to the structure. This compensates the deficiency of k-means, which improve more on datasets with higher overlap (Birch1) than with lower overlap

Table 5 CI and relative CI-values for k-means with different initializations, their repeated k-means variants (100 repeats), RS (5000 iterations) and GA

Dataset	K-means			Repeated KM			RS	GA
	Ste	Ran	Max	Ste	Ran	Max		
CI-values								
A1	6.0	2.5	1.0	4.8	0.4	**0.0**	**0.0**	**0.0**
A2	10.7	4.6	2.6	8.8	1.7	0.4	**0.0**	**0.0**
A3	17.9	6.6	2.9	16.4	3.0	0.6	**0.0**	**0.0**
S1	3.2	1.9	0.7	1.3	0.1	**0.0**	**0.0**	**0.0**
S2	0.6	1.4	1.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
S3	1.2	1.3	0.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
S4	0.4	0.9	1.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Unbalance	4.0	3.9	0.9	3.5	2.4	**0.0**	**0.0**	**0.0**
Birch1	11.3	6.7	5.5	8.6	2.7	2.7	**0.0**	**0.0**
Birch2	75.5	16.6	7.3	74.0	10.8	3.8	**0.0**	**0.0**
Dim32	5.5	3.6	**0.0**	2.7	1.1	**0.0**	**0.0**	**0.0**
Average:	12.4	4.5	2.2	10.9	2.0	0.7	0.0	0.0
Relative CI-values								
A1	30%	12%	6%	24%	2%	**0%**	**0%**	**0%**
A2	31%	13%	7%	25%	5%	1%	**0%**	**0%**
A3	36%	13%	6%	33%	6%	1%	**0%**	**0%**
S1	22%	13%	5%	9%	1%	**0%**	**0%**	**0%**
S2	4%	9%	7%	**0%**	**0%**	**0%**	**0%**	**0%**
S3	8%	9%	5%	**0%**	**0%**	**0%**	**0%**	**0%**
S4	3%	6%	7%	**0%**	**0%**	**0%**	**0%**	**0%**
Unbalance	50%	49%	11%	44%	31%	**0%**	**0%**	**0%**
Birch1	11%	7%	6%	9%	3%	3%	**0%**	**0%**
Birch2	76%	17%	7%	74%	11%	4%	**0%**	**0%**
Dim32	34%	22%	**0%**	17%	7%	**0%**	**0%**	**0%**
Average:	28%	15%	6%	21%	6%	1%	0%	0%

Cases when the correct clustering is always found, are presented in bold

(Birch2); see the results below reported before and after k-means iterations:

	Random initialization		Maxmin initialization	
	Birch1	Birch2	Birch1	Birch2
Initial:	36.6	36.5	21.4	9.6
After k-means:	6.7	16.6	5.5	7.3

Repeated k-means improves all initializations. With the S sets, it almost always determines the correct solution in 100 repeats. However, as the results in Fig. 9 have already demonstrated, RKM cannot solve the problem when there are many clusters. With Birch2, it makes 11% errors with random, and 4% with maxmin initialization. Reference results for the random swap (RS) and the genetic algorithm (GA) provide 0% errors, which indicate that it is possible to find the correct clustering for all these datasets with a single algorithm.

Finally, we made a brief test to determine whether the main observations also hold with two real datasets from the UCI repository. The Leaves dataset [40] contains visual shape features of 100 species of plants, 16 images per species. The Letter dataset [41] contains 26 classes representing letters from A to Z. Each class contains randomly distorted visual features with 20 different fonts.

We further divided the data into 4-5 subsets by clustering so that the clusters with least overlap [21] are in the first subset, clusters with the next least overlap are in the next subset, and so on. Each subset is clustered separately using k-means and compared to the random swap to see how well it succeeds at the clustering task. The results in Table 6 demonstrate that the more overlap, the better k-means performs; the average correlation is 0.78 (Leaves) and 0.88 (Letters).

Table 6 CI-results for the subsets of two real life datasets (Leaves N =71-685 k =20, Letters N =2404-6760 k =6) compared with the corresponding clustering result of the RS algorithm

Dataset	Overlap	K-means			Repeated KM		
		Ste	Ran	Max	Ste	Ran	Max
Leaves	0 %	4.7	4.3	1.6	2.7	2.1	1.0
	0 %	6.3	5.2	2.9	4.6	2.5	2.0
	6 %	6.3	4.0	4.3	4.6	1.5	2.8
	11 %	5.0	3.8	3.2	3.1	1.4	1.9
	38 %	3.3	3.4	3.9	2.1	1.4	2.3
Letters	9 %	0.5	1.0	0.8	0.0	0.0	0.0
	37 %	0.3	0.5	0.2	0.0	0.0	0.0
	55 %	0.8	0.7	0.9	0.0	0.0	0.0
	79 %	0.1	0.2	0.5	0.0	0.0	0.0

6 Conclusions

We have introduced a basic clustering benchmark and proposed using centroid index and success rate to evaluate the results. These two contributions provide a systematic approach to test the performance of the algorithms, and provide a result that is easy to interpret. As a case study, we studied the performance of k-means iterations and reached the following conclusions:

- K-means works better when the clusters overlap. This is the most important factor to predict the success of k-means.
- The more clusters there are, the worse k-means works. The success rate has a linear dependency on the number of clusters: the more clusters there are, the more likely that k-means will fail to solve all clusters correctly.
- Increasing dimensionality does not affect the performance of k-means if the clusters are well separated. With G2 datasets, increasing dimensionality worsens k-means' performance. However, the mediating factor is the overlap, not the dimensionality.
- The performance of k-means is worse when cluster sizes have a strong unbalance. However, the reason has nothing to do with the k-means algorithm itself but it originates from two independent facts: deficiency of the random initialiation and lack of cluster overlap.

Better initialization or repeating k-means can significantly improve it, but they do not completely solve the problems of k-means. If the quality of the clustering is not critical, simply repeating k-means several times is often enough. However, if high clustering accuracy is important, a better algorithm than k-means is recommended.

Finally, we remind that the choice of the algorithm is just one part of the clustering process. The choice of the distance and objective function is much more important in real-life applications. For example, the quality of the clustering was shown not to be critical for the performance of speaker recognition in [60] when any reasonable clustering algorithm was used; including repeated k-means. Random sub-sampling still performed poorly though.

To sum up, clustering results depends primarily on the choice of objective function, and only secondarily on the choice of algorithm. Wrong choice of the function can easily cancel the benefit of a good algorithm and vice versa, a proper objective function can provide better clustering even with worse algorithm. Nevertheless, we have demonstrated conditions when the k-means algorithm is expected to work, and when it can become a bottleneck.

References

1. Forgy E (1965) Cluster analysis of multivariate data: efficiency vs. interpretability of classification. Biometrics 21:768
2. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Berkeley symposium on mathematical statistics and probability, volume 1: statistics. University of California Press, Berkeley, pp 281–297
3. Lloyd SP (1982) Least squares quantization in PCM. IEEE Trans Inf Theory 28(2):129–137
4. Jain AK (2010) Data clustering: 50 years beyond K-means. Pattern Recogn Lett 31:651–666
5. Krishna K, Murty MN (1999) Genetic k-means algorithm. IEEE Trans Syst Man Cybern Part B 29(3):433–439
6. Fränti P (2000) Genetic algorithm with deterministic crossover for vector quantization. Pattern Recogn Lett 21(1):61–68
7. Fränti P, Kivijärvi J (2000) Randomized local search algorithm for the clustering problem. Pattern Anal Applic 3(4):358–369
8. Steinley D, Brusco MJ (2007) Initializing k-means batch clustering: a critical evaluation of several techniques. J Classif 24:99–121
9. Steinbach M, Ertöz L, Kumar V (2003) The challenges of clustering high dimensional data. New Vistas in statistical physics – applications in econophysics, bioinformatics, and pattern recognition. Springer
10. Morissette L, Chartier S (2013) The k-means clustering technique: general considerations and implementation in Mathematica. Tutor Quantitative Methods Psychol 9(1):15–24

11. Liang J, Bai L, Dang C, Cao F (2012) The k-means-type algorithms versus imbalanced data distributions. IEEE Trans Fuzzy Syst 20(4):728–745
12. Luxburg U, Williamson RC, Guyon I (2012) Clustering: science or art. J Mach Learn Res 27:65–79
13. Zhao Q, Fränti P (2014) WB-index: a sum-of-squares based index for cluster validity. Data Knowl Eng 92:77–89
14. Zoubi M, Rawi M (2008) An efficient approach for computing silhouette coefficients. J Comput Sci 4(3):252–255
15. Fränti P, Virmajoki O, Hautamäki V (2006) Fast agglomerative clustering using a k-nearest neighbor graph. IEEE Trans Pattern Anal Mach Intell 28(11):1875–1881
16. Zhang RR, Livny M (1997) BIRCH: a new data clustering algorithm and its applications. Data Min Knowl Disc 1(2):141–182
17. Kärkkäinen I, Fränti P Dynamic local search algorithm for the clustering problem, Research Report A-2002-6
18. Fränti P, Virmajoki O (2006) Iterative shrinking method for clustering problems. Pattern Recogn 39(5):761–765
19. Fränti P, Mariescu-Istodor R, Zhong C (2016) XNN graph. In: IAPR Joint int. workshop on structural, syntactic, and statistical pattern recognition merida, Mexico, LNCS 10029, pp 207–217
20. Rezaei M, Fränti P (2016) Set-matching methods for external cluster validity. IEEE Trans Knowl Data Eng 28(8):2173–2186
21. Maitra R, Melnykov V (2010) Simulating data to study performance of finite mixture modeling and clustering algorithms. J Comput Graph Stat 19(2):354–376
22. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is "nearest neighbor" meaningful. In: Int Conf on database theory, pp 217–235
23. Chavez E, Navarro G (2001) A probabilistic spell for the curse of dimensionality. Workshop on Algorithm Engineering and Experimentation LNCS 2153:147–160
24. Tomasev N, Radovanovi M, Mladeni D, Ivanovi M (2014) The role of hubness in clustering high-dimensional data. IEEE Trans Knowl Data Eng 26(3):739–751
25. Radovanovic M, Nanopoulos A, Ivanovic M (2010) Hubs in space: popular nearest neighbors in high-dimensional data. J Mach Learn Res 11:2487–2531
26. Dasgupta S (2007) The hardness of k-means clustering, Technical Report CS2007-0890. University of California, San Diego
27. Mahajan M, Nimbhorkar P, Varadarajan K (2012) The planar k-means problem is NP-hard. Theor Comput Sci 442:13–21
28. Fränti P, Rezaei M, Zhao Q (2014) Centroid index: cluster level similarity measure. Pattern Recogn 47(9):3034–3045
29. Steinley D, Brusco MJ, Hubert L (2016) The variance of the adjusted rand index. Psychol Methods 21(2):261–272
30. Kanungo T, Mount DM, Netanyahu N, Piatko C, Silverman R, Wu AY (2004) A local search approximation algorithm for k-means clustering. Comput Geom Theory Appl 28:89–112
31. Li SC, Ng YK, Zhang L (2008) A PTAS for the k-consensus structures problem under squared euclidean distance. Algorithms 1(2):43–51
32. Awasthi P, Charikar M, Krishnaswamy R, Sinop AK (2015) The hardness of approximation of euclidean k-means. In: Int. Symp. on computational geometry (SoCG). Eindhoven
33. Arthur D, Vassilvitskii S (2007) K-means++: the advantages of careful seeding. In: ACM-SIAM Symp. on discrete algorithms (SODA), New Orleans, 1027-1035 January
34. Ball GH, Hall DJ (1967) A clustering technique for summarizing multivariate data. Syst Res Behav Sci 12(2):153–155
35. Bradley P, Fayyad U (1998) Refining initial points for k-means clustering. In: Int. Conf. on machine learning. San Francisco, pp 91-99
36. Steinley D (2003) Local optima in k-means clustering: what you don't know may hurt you. Psychol Methods 8:294–304
37. Gonzalez T (1985) Clustering to minimize the maximum intercluster distance. Theor Comput Sci 38(2–3):293–306
38. Tezuka S, Equyer P (1991) Efficient portable combined Tausworthe random number generators. ACM Trans Model Comput Simul 1:99–112
39. Peña J, Lozano JA, Larrañaga P (1999) An empirical comparision of four initialization methods for the *k*-means algorithm. Pattern Recogn Lett 20(10):1027–1040
40. Mallah C, Cope J, Orwell J (2013) Plant leaf classification using probabilistic integration of shape, texture and margin features. Signal Process Pattern Recogn Appl
41. Frey PW, Slate DJ (1991) Letter recognition using Holland-style adaptive classifiers. Mach Learn 6(2):161–182
42. Yan D, Huang L, Jordan MI (2009) Fast approximate spectral clustering. ACM SIGKDD Int Conf on knowledge discovery and data mining, 907–916
43. Xiong H, Wu J, Chen J (2009) K-means clustering versus validation measures: a data distribution perspective. IEEE Trans Syst Man Cybern Part B 39(2):318–331
44. Zhou K, Yang S (2016) Exploring the uniform effect of FCM clustering: a data distribution perspective. Knowl-based Syst 96:76–83
45. Hirsch JE (2005) An index to quantify an individual's scientific research output. PNAS 102(46):16569–72
46. Ben-Hur A, Horn D, Siegelmann HT, Vapnik V (2001) Support vector clustering. J Mach Learn Res 2:125–137
47. Balcan M-F, Blum A, Vempala S (2008) A discriminative framework for clustering via similarity functions. ACM Symposium on theory of computing, 671–680
48. Ackerman M, Ben-David S, Brânzei S, Loker D (2012) Weighted clustering. AAAI Conf on artificial intelligence, 858–863
49. Biçici E, Yuret D (2007) Locally scaled density based clustering. In: Int. Conf. on adaptive and natural computing algorithms. Springer, pp 739–748
50. Pelleg D, Moore AW (2000) X-means: extending k-means with efficient estimation of the number of clusters. Int Conf on machine learning, 1
51. Hinneburg A, Keim DA (1999) Optimal grid-clustering: towards breaking the curse of dimensionality in high-dimensional clustering. Int Conf on very large databases, 506–517
52. Domeniconi C, Gunopulos D, Ma S, Yan B, Al-Razgan M, Papadopoulos D (2007) Locally adaptive metrics for clustering high dimensional data. Data Min Knowl Disc 14(1):63–97
53. Bellman R (1961) Adaptive control processes: a guided tour. Princeton University Press
54. Aggarwal CC, Hinneburg A, Keim DA (2001) On the surprising behavior of distance metrics in high dimensional spaces. In: Int. conf. on database theory, LNCS vol 1973. Springer, pp 420–434
55. Sieranoja S, Fränti P (2018) Random projection for k-means clustering. In: Int. Conf. artificial intelligence and soft computing (ICAISC). Zakopane
56. Hartigan JA, Wong MA (1979) Algorithm AS 136: a k-means clustering algorithm. J R Statist Soc C 28(1):100–108
57. Erisoglu M, Calis N, Sakallioglu S (2011) A new algorithm for initial cluster centers in *k*-means algorithm. Pattern Recogn Lett 32(14):1701–1705

58. Fränti P (2018) Efficiency of random swap clustering. J Big Data 5:13:1–29
59. Duda RO, Hart PE (1973) Pattern classification and scene analysis. Wiley, New York
60. Kinnunen T, Sidoroff I, Tuononen M, Fränti P (2011) Comparison of clustering methods: a case study of text-independent speaker modeling. Pattern Recogn Lett 32(13):1604–1617
61. Huang X, Zhang L, Wang B, Li F, Zhang Z (2018) Feature clustering based support vector machine recursive feature elimination for gene selection. Appl Intell 48:594–607
62. Ultsch A (2005) Clustering with SOM: U*C, workshop on self-organizing maps. Paris, pp 75–82

Pasi Fränti received the MSc and PhD degrees in science from the University of Turku, 1991 and 1994. Since 2000, he has been a professor of computer science at the University of Eastern Finland. He has published 75 journals and 165 peer review conference papers, including 14 IEEE transaction papers. His research interests include clustering algorithms and location based systems.

Sami Sieranoja received the B.Sc. and M.Sc. degrees in University of Eastern Finland, 2014 and 2015. Currently he is a doctoral student at the University of Eastern Finland. His research interests include neighborhood graphs and data clustering.