cs-means: Determining optimal number of clusters based on a level-of-similarity

Rabindra Lamsal¹ · Shubham Katiyar¹

Received: 28 May 2020 / Accepted: 22 September 2020 / Published online: 6 October 2020
© Springer Nature Switzerland AG 2020

Abstract

This paper proposes a centroid-based clustering algorithm, *cs-means*, which is capable of clustering data-points with *n*-features, without having to specify the number of clusters to be formed. The core logic behind the algorithm is a similarity measure that collectively decides whether to assign an incoming data-point to a pre-existing cluster, or create a new cluster and assign the data-point to it. The algorithm is application-specific and applicable when the need is to perform clustering analysis of a stream of data-points, where the similarity measure between an incoming data-point and the cluster to which the data-point is to be associated with, is higher than the predefined level-of-similarity (cluster strictness). The algorithm was experimented on 4 public datasets and 10 isotropic Gaussian blobs. The cluster analysis strongly confirms the objectives of the proposed clustering algorithm.

Keywords Unsupervised · Clustering · Centroid-based.

1 Introduction

The main gist behind clustering is to group data-points into various groups (clusters) based on their features, i.e. properties. The generation of clusters varies application-wise [17], because it depends on what factors are to be taken into consideration to form a particular cluster. But, the focus of every clustering algorithm remains same, i.e. to group similar data-points to a common cluster. The thing that differs is how this goal of forming a cluster is achieved. Different algorithms use different concepts to deal with similarity measure among the data-points. There are many popular clustering algorithms that group data-points based on various strategies to define the similarity measure between them. Centroid-based [9], density-based [7], graph-based [10], etc. are the commonly used strategies.

Often used algorithms like k-means, hierarchical clustering, DBSCAN, etc. require a set of data-points in space,

beforehand. Without making some adjustments to these pre-existing clustering algorithms, it is not possible to cluster a stream of real-time data-points. A real-world problem exists when there is a necessity of grouping a set of data-points but the possible number of clusters the data-points can generate is unknown. Therefore, considering the real-time approach limitation of the algorithms mentioned above, this paper proposes a clustering algorithm that can group incoming data-points without having to initialize the number of clusters to be formed. Hereinafter, the proposed clustering algorithm is termed as *cs-means*, with “cs” standing for “cluster strictness”. The main objectives of the proposed algorithm are to: (1) facilitate those problems which require clustering of data-points based on some predefined level-of-similarity, (2) introduce a real-time approach to a centroid based clustering algorithm, (3) determine the optimal number of clusters for a dataset.

In the literature review, we did not come across any literary works that had exploited the level-of-similarity

✉ Rabindra Lamsal, rabindralamsal@outlook.com; Shubham Katiyar, shubha25_scs@jnu.ac.in | ¹School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi 110067, India.

concept for generating clusters. However, there have been few approaches to generation of clusters of data streams. D-Stream [5] used a density-based approach to generate and adjust clusters in realtime. [1] divided the whole clustering process into two components: online and offline. Online component periodically stored detailed summary statistics and offline component facilitated an analyst to use variety of inputs including time horizon and number of clusters to be formed. Work in [8] maintained clustering of a sequence of data-points observed at a period of time. In [3], based on the work of [2], the authors proposed an algorithm for real-time data analytics that works based on the eccentricity of each sample to all existing data and existing clusters. Similarly, some notable works have been done in the area of determining optimal numbers of clusters. In [15], the authors use U-Control Chart (UCC) method with Automatic Clustering using Differential Evolution (ACDE) [6, 11] to determine the number of clusters in k-means. [13] presented the Spectral Global Silhouette method, to calculate the optimal number of clusters in a dataset using a Spectral Clustering algorithm. In [16] outliers were detected and removed to better estimate the optimal number of clusters when there are more variables than objects. In [12], a cluster validity evaluation method based on the ratio of deviation of sum-of-squares and Euclidean distance was proposed, which determines the near-optimal number of clusters.

The paper is organized as follows. Section 2 describes the core logic of the proposed algorithm. Implementation of the algorithm is done in Sect. 3. Section 4 discusses the experimental results and Sect. 5 concludes the paper.

2 The `cs-means` algorithm

The `cs-means` algorithm requires an initial declaration of cluster strictness. Cluster strictness is the lowest permitted level-of-similarity between a data-point and the centroid of a cluster. Here, cluster centroid is the average of features of the data-points present in a cluster and is an effective way of representing a particular cluster. If cluster strictness is set to a measure of 60, then the accepted variability between a cluster centroid (C) and a data-point (D) should be a measure of 40 at-most, for D to be associated to cluster C.

Hence, the `cs-means` algorithm is capable of generating clusters based on a level-of-similarity. If one needs clusters with very low variance among the data-points, the cluster strictness can be adjusted accordingly; maybe around 80-95%. That is why cluster strictness plays a significant role in generating clusters, and its value depends entirely on what application the clustering is being done.

Algorithm 1, is the pseudo-code for the proposed `cs-means` clustering algorithm.

2.1 Data-point and features

A data-point can have multiple features (n-features). Those features are the characteristics which collectively define the data-point. A stream of data-points can be grouped into various clusters, based on their features. Below is an example of a data-point with 7 features.

A data-point : 5 10 15 20 25 30 35

During the implementation of the `cs-means` algorithm, the above pattern is used to represent a data-point, and similarity measure between the data-points and cluster(s) is calculated accordingly. The algorithm takes data-points with n-features, one data-point at a time, and requires all the data-points to have the same fixed number of features. When the algorithm runs for the very first time and is waiting for a new data-point; the number of cluster(s) is 0. Cluster strictness needs to be defined beforehand. Based on the requirements, this value can be adjusted. Higher the value of cluster strictness, less is the variance between the data-points in a cluster. Depending on the data-points, using a higher value of cluster strictness result in a large number of clusters. This is more likely to happen if the incoming data-points are less identical to each other.

After assigning some value to cluster strictness, say 70%, the number of features that should be matched between a data-point and a cluster is calculated using Formula 1.

$$\text{should_match_features} = \frac{\text{no_of_features} * \text{cluster_strictness}}{100} \quad (1)$$

Suppose, the data-points that are to be clustered have 20 features. That means cluster strictness with 70% results into 14 features that must be matched (at least) for a data-point to be associated with a particular cluster.

2.2 Qualifying feature

A feature qualifies to be counted as a matched feature when its similarity measure lies between the range: (cluster_strictness) and (100 + (100 – cluster_strictness)). That is, the valid range is 70–130 when cluster strictness is considered 70. Basic mathematics can be used when variability check of 5 and 13 is to be done with respect to 9. 9 with respect to 9 gives a similarity measure of 100. 5 with respect to 9 gives a similarity measure of 55.56. And, 13 with respect to 9 gives a similarity measure of 144.44. If allowed variability is considered to be a measure of 50, then similarity measure of both 5 and 13 falls under the range 50–150, and are considered to be associated to 9 by at-least a measure of 50.

In the case of data-points, Formula 2 is used to calculate the similarity measure.

$$S(C_i, F_j) = \frac{100 * datapoint_j}{centroid(i, j)} \quad (2)$$

where i is an index for clusters, and j is an index for features.

As an example, in Formula 2, $i = 3$ and $j = 7$ reflect the fact that cluster 3 is being considered, and 7th feature of a data-point and 7th feature of the centroid of cluster 3 are being used to compute the level-of-similarity between them. If the resulted similarity measure lies between the range (cluster_strictness)

and $(100 + (100 - \text{cluster_strictness}))$, then the 7th feature is said to be matched, and the value of matched feature counter for the cluster 3 is incremented by 1. This process of calculating the similarity feature of an incoming data-point is done with all the cluster(s). And if the matched feature counter for a cluster reaches the limit of the minimum number of features that should be matched between a data-point and a cluster, that particular cluster is now placed into a list of qualified clusters.

Algorithm 1: The cs-means Algorithm

```

1  Declare cluster_strictness.
2  Initialize cluster_counter ← 0.
3  Calculate should_match_features using formula.
4  while data-points do
5      Read a data-point D
6      if cluster_counter = 0 then
7          Create a new cluster Ccluster_counter.
8          Increment cluster_counter by 1.
9          Assign D to the newly formed cluster.
10         D be the centroid of the cluster.
11     else
12         for all clusters Ci, where i ranges from 1 to cluster_counter do
13             for all features Fj, where j ranges from 1 to no_of_features do
14                 Calculate similarity measure using formula.
15                 if S(Ci, Fj) ≥ (cluster_strictness)
16                     and S(Ci, Fj) ≤ (100 + (100 - cluster_strictness)) then
17                     | Increment matched_featuresi by 1.
18                 end
19             end
20             if matched_featuresi ≥ should_match_features then
21                 | Add Ci to the list of qualified_clusters
22             end
23         end
24         if qualified_cluster list is empty then
25             Create a new cluster Ccluster_counter.
26             Increment cluster_counter by 1.
27             Assign D to the newly formed cluster.
28             D be the centroid of the cluster.
29         end
30         if qualified_cluster list contains single cluster then
31             Assign D to the cluster.
32             Re-calculate the centroid.
33         else
34             Identify cluster(s) with maximum matched_features.
35             if single cluster arises then
36                 Assign D to the cluster.
37                 Re-calculate the centroid.
38             else
39                 Identify the cluster with max average of qualifying similarity
40                 measures.
41                 Assign D to the cluster.
42                 Re-calculate the centroid.
43             end
44         end
45     end
  
```

2.3 Qualified cluster list

This list contains the cluster(s) that satisfy the condition of having at least a minimum number of features that have matched with an incoming data-point. After calculation of similarity measure for a data-point with all the cluster(s), there might arise any one of the following three situations.

2.3.1 A qualified cluster list is empty

The qualified list being empty indicates that no similar cluster(s) were found within the provided level-of-similarity. Hence, a new cluster is generated, and the data-point is assigned to the newly formed cluster. The data-point is the centroid of this cluster.

2.3.2 A qualified cluster list contains exactly 1 cluster

In this case, the data-point is simply assigned to the cluster which is present in the list. And, the centroid of the cluster is re-calculated.

2.3.3 A qualified cluster list contains more than 1 cluster

Case 1: If the list contains more than 1 cluster, the cluster with maximum matched features is identified. The data-point is now assigned to the identified cluster. Case 2: Sometimes, two or more cluster might come-up with the same highest number of matched features. In this case, the cluster with a maximum average of qualifying similarity measures is identified, and the data-point is assigned to that cluster accordingly.

2.4 Conflicting clusters

Whenever multiple clusters show up in qualified cluster list, while trying to identify the nearest similar cluster based on the highest number of matched features, those clusters are said to be the conflicting ones. In such case, an average of qualifying similarity measures is calculated for all the clusters which are in the qualified list and have the same number of matched features, and finally, the cluster with the maximum average is considered to be the cluster to which the data-point is associated. While calculating an average, only the qualifying similarity measures are being considered. One fact is also being taken into consideration that technically 60 and 140 are same with respect to 100. Both have a variability measure of 40.

3 Implementation of the c_s -means algorithm

Let us consider the following six data-points, each of 10 features, that are to be clustered. The data-points are input to the algorithm in a serial fashion.

Data-point 1 : 10 15 20 25 30 35 40 45 50 55

Data-point 2 : 09 35 18 45 10 32 60 41 10 20

Data-point 3 : 18 13 18 27 30 38 38 41 49 57

Data-point 4 : 20 20 18 05 15 34 50 43 10 50

Data-point 5 : 17 17 18 15 22 35 44 43 10 53

Data-point 6 : 10 32 20 45 12 55 40 55 09 25

Let us suppose, level-of-similarity for a cluster is considered to be 60. That means variability measure of 40 between a data-point and a cluster centroid is acceptable. With cluster strictness set to 60, using formula 1 we get minimum number features that must be matched to be 6.

Initially, when Data-point 1 is input to the algorithm, because of the absence of cluster(s), a new cluster C1 is created, and Data-point 1 is assigned to it, and features of Data-point 1 is assumed to be the centroid of C1.

C1 = Data-point 1

C1 : 10 15 20 25 30 35 40 45 50 55

For Data-point 2

S(C1,Fi)	90	233.34	90	180	33.34	91.43	150	91.11	20	36.36
----------	----	--------	----	-----	-------	-------	-----	-------	----	-------

The number of qualified features is 4, which is less than 6. C1 cannot be added to the list of qualified clusters. Since there are no clusters in the list of qualified clusters, a new cluster C2 is generated, and Data-point 2 is assigned to C2.

C2 = Data-point2

C2 : 09 35 18 45 10 32 60 41 10 20

For Data-point 3

S(C1,Fi)	180	86.67	90	108	100	108.57	95	91.11	98	103.64
S(C2,Fi)	200	37.14	100	60	300	118.75	63.33	100	490	285

Only C1 qualified to be in in the list of qualified clusters, Data-point 3 is assigned to C1.

C1 = Data-point1, Data-point3

New C1 : 14 14 19 26 30 36.5 39 43 49.5 56

For Data-point 4

S(C1,Fi)	142.86	142.86	94.74	19.23	50	93.15	128.21	100	20.20	89.29
S(C2,Fi)	222.22	57.14	100	11.11	150	106.25	83.33	104.87	100	250

Since there no cluster qualified to be kept in the list of qualified clusters, a new cluster C3 is generated, and Data-point 4 is assigned to C3.

C3 = Data-point4

New C3 : 20 20 18 05 15 34 50 43 10 50

For Data-point 5

S(C1,Fi)	121.43	121.43	94.74	57.59	73.33	95.89	112.82	100	20.20	94.64
S(C2,Fi)	188.89	48.57	100	33.33	220	109.38	73.33	104.88	100	265
S(C3,Fi)	85	85	100	300	146.67	102.94	88	100	100	106

Since C1 and C3 qualified to be in the list of qualified clusters and both the clusters have the same number of qualified features. Now, the average of the qualifying features is calculated for both the clusters. Qualifying features beyond 100 are scaled below 100 using Formula 3:

$$\text{scaled value} = 100 - (\text{value beyond hundred} - 100) \quad (3)$$

Average of qualifying features (C1) = 87.87

Average of qualifying features (C3) = 93.63

Since the average of qualifying features for C3 is higher, Data-point 5 is assigned to C3.

C3 = Data-point 4, Data-point 5

New C3 : 18.5 18.5 18 10 18.5 34.5 47 43 10 51.5

For Data-point 6

S(C1,Fi)	71.43	228.57	105.26	173.08	40	150.68	102.56	127.91	18.18	44.64
S(C2,Fi)	111.11	91.43	111.11	100	120	171.88	66.67	134.15	90	125
S(C3,Fi)	54.05	172.97	111.11	450	64.86	159.42	85.11	127.91	90	48.54

Since C2 is the only cluster qualifying to be in the list of qualified clusters, Data-point 6 is assigned to C2.

C2 = Data - point 2, Data - point 6

New C2 : 9.5 33.5 19 45 11 43.5 50 48 9.5 22.5

The data-points were successfully grouped into 3 various clusters based on the similarity measure of their features. Cluster strictness of 60 was considered, hence permitted variability measure between a data-point and a cluster centroid was 40.

4 Experimental results

The `cs-means` algorithm was experimented on 14 different datasets for evaluation purposes. Among those datasets, 4 of them are popularly used public datasets [4], namely Iris Dataset, Wine Dataset, Wisconsin Diagnostic Breast Cancer (WDBC) Dataset, and Wisconsin Prognostic Breast Cancer (WPBC) Dataset. The remaining 10 datasets are the randomly generated isotropic Gaussian blobs of varying number of samples, features, and centers. The

scikit-learn library [14] was used to create those Gaussian blobs. The overview of the datasets used for experimentation is given in Table 1 and Table 2.

The 4 public datasets are generally associated with the classification task. The different classes within these datasets have been considered their respective centers in this experimentation for clustering task. No instances of these datasets have been removed except for the WPBC Dataset, which had a single unknown value in four of its samples. Therefore, the total number of samples in the WPBC Dataset was 194 after removing those four samples.

For cluster analysis, the intra-cluster distance was computed for all the datasets against the discrete values of the cluster strictness ranging from 5 to 95% with a 5% difference gap. Euclidean distance was used as the distance function for computing the total intra-cluster distance. For fair experimentation, each Gaussian blob has

been generated with varying sizes, features, and centers. Figure 1 shows the results obtained from the 4 public datasets. Similarly, Fig. 2 shows the results obtained from the 10 isotropic Gaussian blobs. Each experimental result graph illustrates the effect of varying the value of cluster strictness on the formation of clusters of data-points. The blue plot represents the number of clusters formed against a value of cluster strictness. The red plot represents the total intra-cluster distance against a value of cluster strictness.

4.1 Discussion

The experimental results strongly confirm the main objectives of the `cs-means` algorithm. Figure 1 shows the results obtained from the real-world public datasets. As the cluster strictness is increased, the number of clusters also starts increasing. It is also evident that with the increase in the number of clusters, the intra-cluster distance improves. In those 4 public datasets, only a single cluster is formed until the value of cluster strictness reaches beyond a specific limit. In the Iris Dataset, there are three classes, and the clusters for these three classes

Table 1 Overview of the 4 Public Datasets

SN.	Dataset	#samples	#features	#centers
1	Iris Dataset	150	4	3 (classes)
2	Wine Dataset	178	13	3 (classes)
3	WDBC Dataset	569	30	2 (classes)
4	WPBC Dataset	194	33	2 (classes)

Table 2 Overview of the 10 Isotropic Gaussian Blobs

SN.	Dataset	#samples	#features	#centers
1	s1000_f100_c10 Dataset	1000	100	10
2	s1000_f100_c50 Dataset	1000	100	50
3	s5000_f50_c10 Dataset	5000	50	10
4	s5000_f50_c50 Dataset	5000	50	50
5	s1000_f500_c10 Dataset	1000	500	10
6	s1000_f500_c50 Dataset	1000	500	50
7	s500_f1000_c10 Dataset	500	1000	10
8	s500_f2000_c50 Dataset	500	2000	50
9	s25000_f20_c50 Dataset	25,000	20	50
10	s50000_f20_c50 Dataset	50,000	20	50

seem to be getting formed only after the cluster strictness is set beyond 65%. And with the increase in the strictness value, the number of clusters also seems to be increasing. The same is the case for the other 3 public datasets as well. The additional clusters seem to be getting formed for Wine Dataset, WDBC Dataset, and WPBC Dataset after reaching 60%, 40%, and 50% of cluster strictness, respectively.

Similarly, the *cs-means* algorithm was used on 10 isotropic Gaussian blobs. Figure 2 shows the results obtained from those blobs. The s1000_f100_c10 Dataset and s1000_f100_c50 Dataset had 1000 samples with 100 features, but 10 and 50 centers, respectively. For the former Dataset, the optimal number of clusters showed up when the cluster strictness was in the range 40–65%, and for the latter, the range was 40–65%. Similarly, the number of samples was increased, and the number of features was brought to 50 for generating the s5000_f50_c10 Dataset and s5000_f50_c50 Dataset, but with 10 and 50 centers, respectively. In the case of these datasets, the former had the optimal number of clusters when the cluster strictness was in the range 40–60%, and for the latter, the range was 45–60%. Similarly, for all the Gaussian blobs, the optimal number of clusters seem to be appearing with cluster strictness in the range $(50 \pm 15)\%$. This condition holds for the public datasets as well. Interestingly, in the case of large samples, s25000_f20_c50 Dataset and s50000_f20_c50 Dataset, the range of cluster strictness for obtaining the optimal number of clusters was too narrow and was seen to be converging at 50%. Summing up, the cluster analysis suggests that the initial lookout value for cluster strictness can be supposed as 50% while using the *cs-means* algorithm. Besides, the strictness value can be adjusted accordingly if the task is to group data-points based on allowed proximity of variance within a cluster.

Interestingly two significant deductions can be made from the experimental results. One, the *cs-means* algorithm seems to be abundantly utilizing the value of cluster strictness in the formation of clusters. Two, the elbow

Fig. 1 Results on the 4 Public Datasets

Fig. 2 Results on the 10 Isotropic Gaussian Blobs

of the intra-cluster distance curve can be considered as an ideal location for determining the optimal number of clusters for a dataset. Besides these two deductions, the `cs-means` algorithm's real-time approach is a significant improvement over traditional clustering algorithms. Algorithms such as K-Means, hierarchical clustering and DBSCAN require all the data-points that are to be grouped, to be already in the vector space. However, the algorithm proposed in this paper takes-in data-points in a serial manner. This real-time behavior contributes to the clustering of the data-points that are yet unseen, but will eventually be available.

5 Conclusion

In this paper, a clustering algorithm, `cs-means`, was proposed, which can cluster data-points without having to specify the number of clusters to be formed. The only hyper-parameter in the algorithm is cluster strictness—the lowest permitted level-of-similarity between a data-point

and the centroid of a cluster. The algorithm's primary benefit is that it does not necessarily require all the data-points to be available in the vector space.

In the initial part, the theoretical aspect of the `cs-means` algorithm was discussed and implemented on 6 data-points (all of them with 10 dimensions) with the supposition that the data-points are input to the algorithm one after another, just like a stream-of-data in a serial fashion. The experimental results showed that the formation of clusters can be manipulated by increasing or decreasing the value of cluster strictness. It was also evident that cluster-strictness can help identify the optimal number of clusters for a dataset based on the elbow of the intra-cluster Euclidean distance curve. Summing up, the algorithm is applicable in a real-world scenario where the task is to group a stream of data-points, incoming to a system, based on their similarity with the average of features (centroid) of data-points existing in their respective previously formed clusters. If none of the existing clusters satisfy the similarity measure for a new data-point, a new

Fig. 2 (continued)

cluster is formed, and the data-point is assigned to the newly formed cluster.

5.1 Future directions

When the number of samples and the features are large, the cluster formation becomes time-intensive. Therefore, it can be a new problem, from here, to decrease the number of similarity checks done when new data-points arrive. Instead of checking the similarity of data-points with all the pre-existing clusters, it would reduce the required number of computations if the algorithm can be adjusted to consider only a certain number of clusters for similarity checks.

Compliance with ethical standards

Conflicts of interest The authors declare that they have no conflict of interest.

References

- Aggarwal CC, Philip SY, Han J, Wang J (2003) A framework for clustering evolving data streams. In: Proceedings 2003 VLDB conference. Elsevier, pp 81–92
- Angelov P (2014) Anomaly detection based on eccentricity analysis. In: 2014 IEEE symposium on evolving and autonomous learning systems (EALS). IEEE, pp 1–8
- Bezerra CG, Costa BSJ, Guedes LA, Angelov PP (2016) A new evolving clustering algorithm for online data streams. In: 2016 IEEE conference on evolving and adaptive intelligent systems (EALS). IEEE, pp 162–168
- Blake CL, Merz CJ (1998) Uci repository of machine learning databases
- Chen Y, Tu L (2007) Density-based clustering for real-time stream data. In: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 133–142
- Das S, Abraham A, Konar A (2007) Automatic clustering using an improved differential evolution algorithm. *IEEE Trans Syst Man Cybern Part A Syst Hum* 38(1):218–237
- Ester M, Kriegel HP, Sander J, Xu X et al (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Kdd* 96:226–231
- Guha S, Mishra N, Motwani R, o'Callaghan L (2000) Clustering data streams. In: Proceedings 41st annual symposium on foundations of computer science. IEEE, pp 359–366 (2000)
- Hartigan JA, Wong MA (1979) Algorithm as 136: a k-means clustering algorithm. *J R Stat Soc Ser C (applied statistics)* 28(1):100–108
- Hartuv E, Shamir R (2000) A clustering algorithm based on graph connectivity. *Inf Process Lett* 76(4–6):175–181
- Kuo R, Suryani E, Yasid A (2013) Automatic clustering combining differential evolution algorithm and k-means algorithm. In: Proceedings of the institute of industrial engineers asian conference. Springer, pp 1207–1215
- Li X, Liang W, Zhang X, Qing S, Chang PC (2020) A cluster validity evaluation method for dynamically determining the near-optimal number of clusters. *Soft Comput* 24(12):9227–9241
- Mur A, Dormido R, Duro N, Dormido-Canto S, Vega J (2016) Determination of the optimal number of clusters using a spectral clustering optimization. *Expert Syst Appl* 65:304–314
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830

15. Viloria A, Lezama OBP (2019) Improvements for determining the number of clusters in k-means for innovation databases in SMES. *Proc Comput Sci* 151:1201–1206
16. Wang M, Abrams ZB, Kornblau SM, Coombes KR (2018) Thresher: determining the number of clusters while removing outliers. *BMC Bioinform* 19(1):9
17. Xu R, Wunsch D (2005) Survey of clustering algorithms. *IEEE Trans Neural Netw* 16(3):645–678

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.