



Intermediate Information Grouping in Cluster Recognition

Chloe Chun-wing Lo¹(✉), Markus Hollander², Freda Wan³,
Alexis-Walid Ahmed³, Nikki Bernobić³, Nick Nuon³,
and Michael Shrider³

¹ Cherrypicks Limited, Kowloon, Hong Kong
chloelcw@cherrypicks.com

² Center for Bioinformatics, Saarland University, Saarbrücken, Germany

³ Sirius18, New York, USA

Abstract. Unsupervised clustering takes human brains a split second to complete in 2D space, while existing algorithms require many iterations involving all data points and an initial number of clusters ‘k’ to provide meaningful results. This initial ‘k’ cannot be provided by human if the data is in higher dimension where visualization is practically impossible. Attempts to calculate this value have low performance and give ambiguous results which are unhelpful to human judgment. This presents great motivation to search for a method to provide that initial ‘k’ in higher dimension. Human brains naturally group things in proximity together. By imitating this process and creating a middle process of grouping the data into subregions and mapping the data in each region into a bitmap of data densities, estimating the centroid locations and number of clusters can be simplified into a process of local maxima detection. A run of the algorithm on 2D data proved that it was effective for data with Gaussian-like distribution with some tolerance to overlapping. The algorithm therefore has great potential for data of higher dimension after generalization. This algorithm gives unambiguous initial ‘k’ and fairly accurate estimation of centroids which can speed up various popular clustering algorithms, including the k-means and Gaussian mixture models. Future research on middle grouping processes in human cognition, which may prove valuable in providing better machine learning algorithms, are also called for.

Keywords: Clustering · Proximity grouping · Centroid estimation

1 Literature Review

Clustering is one of the basic data science tasks, where the goal is to separate unlabeled datasets into groups so that data points within a group have more in common in between them than with data points outside of it. This definition highlights typical non-trivial problems a good clustering algorithm must be able to handle, for example:

- What is a cluster?
- How many clusters are there?

- What is the definition of “similarity”?
- Where does a cluster end?
- What shape can a cluster have?

Such challenges have led to a very wide variety of approaches [1], making a comprehensive summary difficult to establish, but most obvious families of clustering algorithms are defined according to the metric/method they use to evaluate differences between points [2, 3]. A reminder of the most commonly seen unsupervised algorithms is provided:

- **Distribution based clustering methods** start by assuming an underlying statistical distribution and will then try to maximize the fit to the data by tuning parameters of the generating function; the initialization of such algorithms is a critical phase that sometimes hinders repeatability as they are prone to getting stuck in local minima, and the choice of the distribution is often a gamble. One of the most successful examples would be Gaussian Mixture Models.
- **Hierarchical clustering methods** attempt to either start from the assumption that all points are in the same big cluster then separate them progressively into a constellation of point-like clusters while keeping track of the order in which subgroups secede; alternatively, they may start from the bottom and successively merge all clusters into one while keeping track of the assimilations; either way, hierarchical clustering is based on a measure of distance (generally Euclidean or Manhattan) between points and are thus sensitive to outliers with an inability to correct themselves during their execution.
- **Centroid-based clustering methods**, largely based on one of the most widely-used clustering algorithm, the K-means [4], attempt to find clusters of data by calculating a centroid which is not necessarily part of that dataset; the choice of the way centroids are arranged is determined by a cost function, however the choice of the number of centroids itself is still a largely unresolved question, often requiring user input which can be absent or unreliable in high dimensional and/or messy data, or using methods adapted for narrow cases. Since that family needs an a priori number of clusters to start, dozens of techniques have been devised to address that blind spot, called cluster validation techniques that are generally broken down into three different kinds: internal, external, and relative [5]. Internal methods describe measures that depend only on the data used in the clustering itself and the clustering schema, while external methods rely on additional information, for example, a user specific intuition. Relative methods consist in running different clustering schemas with different parameters then selecting the «best» one according to a validity index or predefined criterion. One such popular method with the K-mean is the elbow method, where the clustering is done with different K then some measure of error applied. When adding another cluster K_{n+1} stops improving performance, K_n is found to be optimal. All these methods have in common their computational cost, and the need to use a few of them in parallel to offset their respective biases and assumptions, leading to the circulation of «recipes» of these validation techniques with various popularities.

- **Density based clustering methods** [6] define clusters by a measure of the local density in data points, with the assumption that boundaries will tend to have a low density largely due to noise; such methods thus handle outliers and noise quite well since they generally cannot compete in quantity and density with more plausible clusters in a given representative dataset; the major difficulty in those methods is that a predefined reach is often used to prevent overfitting, so that flexibility of the algorithm with regard to the scales it can simultaneously resolve is limited, and detecting boundaries relies on detecting a drop in density, which can be difficult to ascertain and is very dependent on the consistency across clusters. At the frontiers with Topological Data Analysis and Artificial Neural Networks, approaches such as t-SNE [7] or self-organizing maps (SOM or Kohonen Maps [8]) have been used because they generally allow for topology preserving dimensionality reduction of datasets while making even high-dimensional clusters more apparent to the naked eye; such methods can however be thought of as extended versions of previously described algorithms, SOMs in particular have been shown to be rigorously identical in their late steps to K-means [9].
- **Bioinspired algorithms**, among which herd algorithms are particularly promising [10], characterized by a tendency for decentralized processing relying on emergent clustering, have been steadily developed since the classical ant colony inspired algorithm was first laid out; they are robust to noise and dynamic, making them suitable for time-dependent data analysis [11], whereas previous techniques are generally confined to static data (although recent developments are starting to change that [12]).

However, none of the above seems to truly parallel with the performance of human brain. Humans are able to detect cluster of objects by proximity grouping [13] in split second. Since data points in close proximity suggest a denser data points relative to their surroundings, it is of great interest whether a local density maxima could successfully detect the number of clusters with a better efficiency and performance. An algorithm that first groups the data into sub regions and comparing densities in each region considering only their neighboring regions is then developed in the hope of providing a more accurate and reliable estimation of the number of clusters in a given dataset, and also possibly the estimation of the location of their centroids.

2 Procedure

We tested our algorithm on all datasets from <https://cs.joensuu.fi/sipu/datasets/>. Here, we are showcasing the following 4 datasets: S-sets S-3 and S-4, A-set A-3 and Birchset Birch 1.

2.1 Hardware and Software

The algorithm is run on a MacBook Pro (Retina, 13-in., late 2013), Intel Core Duo i5, 8 GB 1600 MHz DDR3 RAM, macOS High Sierra version 10.13.2 using Python3 v3.6.3 with Seaborn v0.8.1, Pandas v0.21.0 and NumPy v1.13.3.

2.2 Algorithm

The algorithm consists of two parts. First, it searches for the best grid size to divide the dataset into subregions, and then scans for local maxima.

Determination of Optimal Grid Size. In the following passages, the letter m denotes how many portions each side of the bounding box of the data is divided into (which results in m^2 grid boxes in total). The computation of the best grid size is done as follows:

1. Set the initial m is set to be the box length divided by 2.
2. The bounding box of the data is segmented into m^2 grid boxes.
3. The number of data points present in each grid box is calculated.
4. Check whether the number of boxes with single data is less than the tolerance level (set at 0.5% in this paper) of all data points.
5. If the above criteria is not met, m will be decreased exponentially by multiplying the adjustment factor (set at 0.0975 in this paper).
6. The above process repeats until the criteria in 4 is met.

Identification of Local Maxima. The local maxima are identified as follows:

1. Go through all the grid boxes from the top-left to bottom-right by a 3-by-3 moving window.
2. Mark the coordinates of the centre of the centre box in the window as a centroid of a cluster if it has the maximum number of data points compared with the rest.

After the above process, both the number of clusters and the estimated location of the centroids are obtained.

Pseudocode and Complexity.

```

# PREPROCESSING
let N = number of data points          # assignment, O(1)
let t = tolerance level, a fraction    # assignment, O(1)
let r = decrease factor, between 0 and 1
                                          # assignment, O(1)

Find max and min of the data set in both dimensions
      # finding the max two times and
      # min two times, 4*O(N)

Get bounding box by comparing max and min for both dimen-
sions
      # assigning max - min twice, 2*O(1)

let M = number of pixels of the longer side of the bound-
ing box
      # finding the larger number among
      # 2 numbers, O(2)

# FINDING OPTIMAL GRID SIZE
Let m = M                               # assignment, O(1)

DO:
/* DO-WHILE LOOP:
  number of iterations for m to reach a grid-size of 1
  (absolute worse case) given a predetermined decrease
  factor r, O(ln(M)/-ln(r))
  */
  m = m*r                                # assignment, O(1)
  Map number of data points into a m*m array by divid-
  ing the side of bounding box by m
      # going through each data point, O(N)
  Check number of grid boxes with only one data point
      # going through each grid box, O(M^2)
  WHILE (number of boxes containing only a single data
  point is under N*t)  # boolean checking, O(1)

# FINDING LOCAL DENSITY MAXIMA
For each grid box from the top-left to bottom-right by a
3*3 moving window:
      # going through each grid box, O(M^2)
  check maximum number of data points among the 9 grid
  boxes in the window  # going through 9 boxes, O(9)
  Check whether the centre box has the maximum number
  of data points      # boolean checking, O(1)
  Store the box if yes  # assignment, O(1)
Count number of identified local maxima
      # going through all stored local
      # maxima, worse case O(M^2)
Return the number of local maxima and their coordinates
      # termination of algorithm

```

The number of iterations of the DO-WHILE loop can be derived by solving for x :

$$\begin{aligned}
 Mr^x &\leq 1 \\
 \ln(N) + x\ln(r) &\leq 0 \\
 x\ln(r) &\leq -\ln(M) \\
 x &\geq -\frac{\ln(M)}{\ln(r)}
 \end{aligned} \tag{1}$$

As $0 < r < 1$, $\ln(r)$ would be negative. So the inequality sign is flipped in the last line in Eq. (1).

For processes that involved the grid boxes, the worse case was assumed to be the smallest grid size of side 1 which gave M^2 number of steps.

The overall complexity will be:

$$\begin{aligned}
 &O\left(3 + 4N + 5 + \frac{\ln(M)}{-\ln(r)}[1 + N + M^2 + 1] + M^2(11) + M^2\right) \\
 = &O\left(\frac{\ln(M)}{-\ln(r)}[N + M^2 + 2] + 11M^2 + M^2 + 4N + 9\right) \\
 = &O\left(\frac{\ln(M)}{-\ln(r)}[N + M^2 + 2] + 12M^2 + 4N + 9\right)
 \end{aligned} \tag{2}$$

Since r is a predefined value that can be fixed across different datasets, it is considered to be a constant. $\ln(r)$ is negative and therefore $-\ln(r)$ can be dropped altogether as a positive constant. After dropping constants,

$$\begin{aligned}
 &= O(\ln(M)[N + M^2] + M^2 + 4N) \\
 &= O(N\ln(M) + M^2\ln(M) + M^2 + 4N) \\
 &= O(N[\ln(M) + 4] + M^2[\ln(M) + 1]) \\
 &= O(N\ln(M) + M^2\ln(M))
 \end{aligned} \tag{3}$$

3 Results

See Figs. 1, 2, 3, 4 and Table 1.

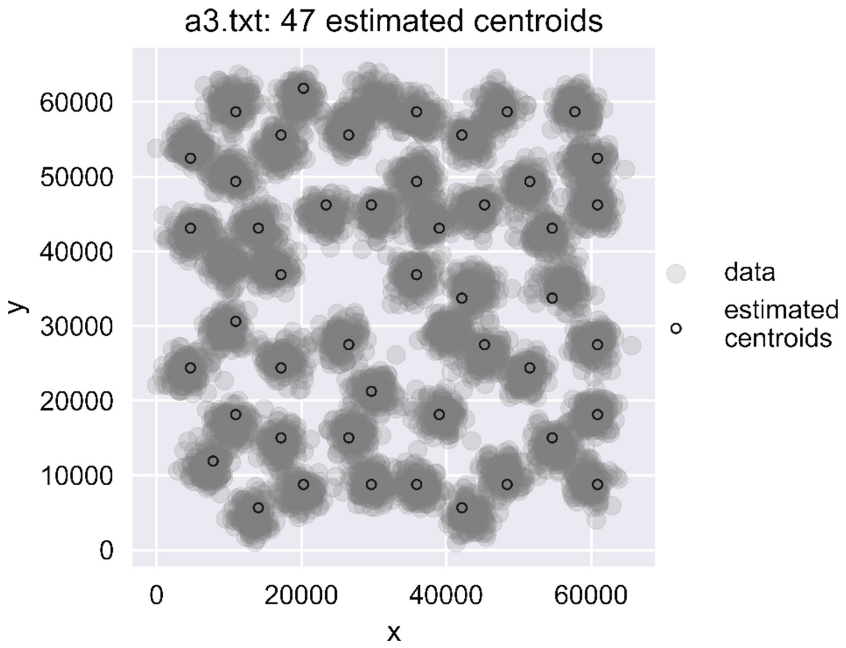


Fig. 1. Application of algorithm on a3 dataset

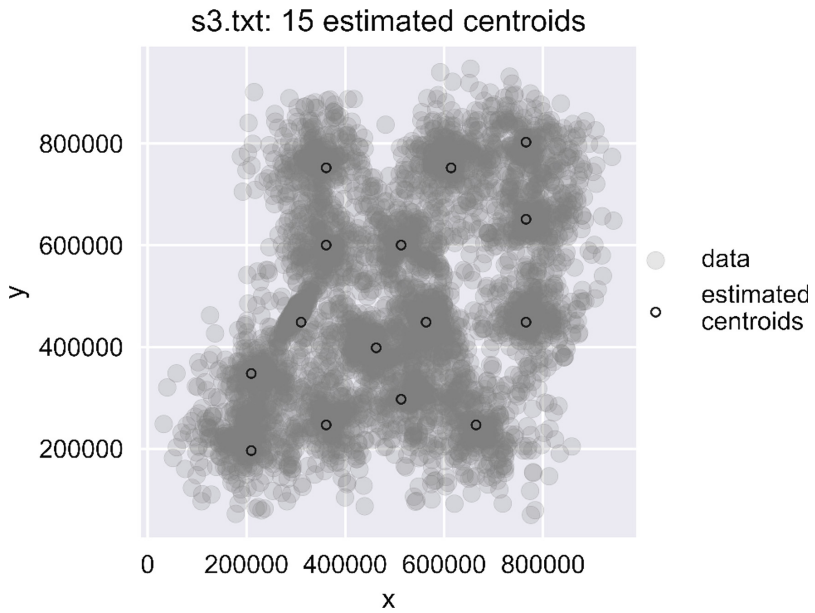


Fig. 2. Application of algorithm on s3 dataset

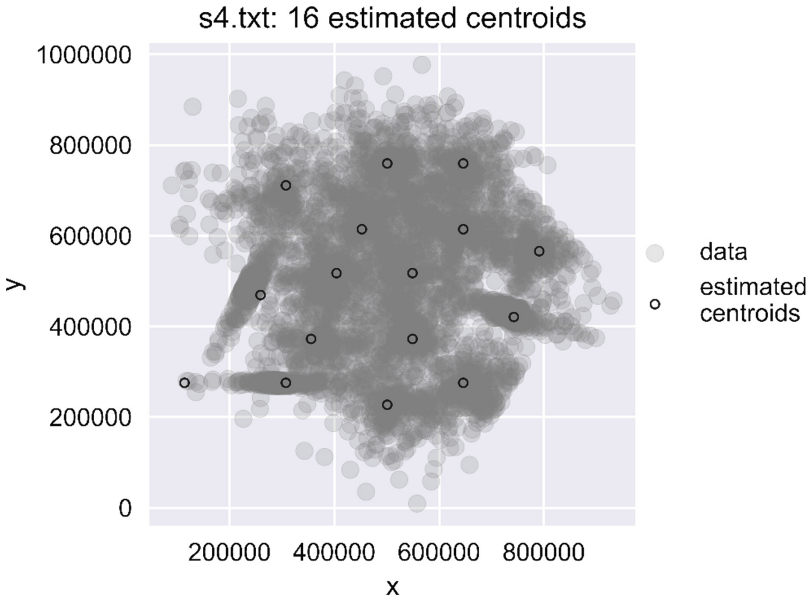


Fig. 3. Application of algorithm on s4 dataset

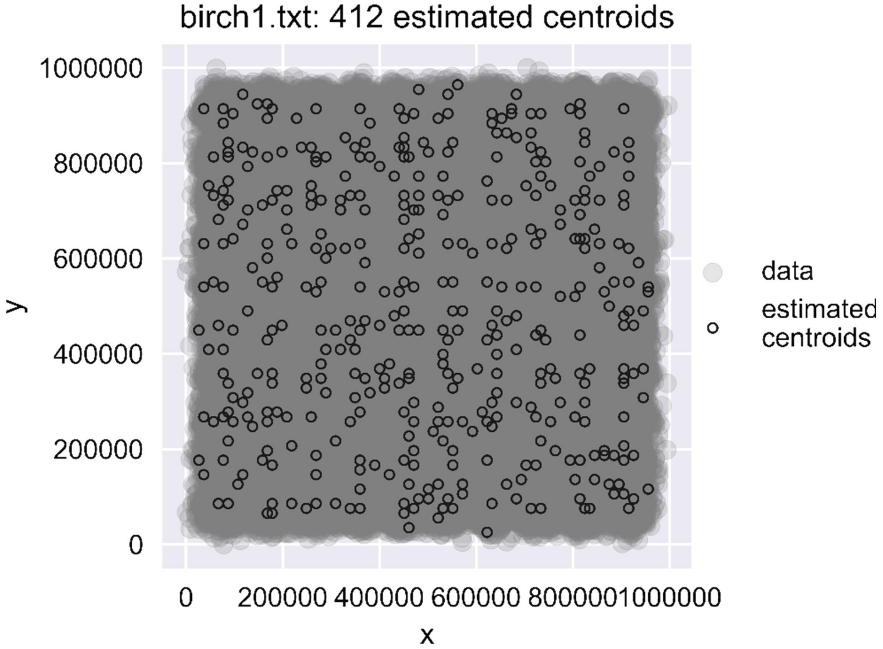


Fig. 4. Application of algorithm on Birch1 datasets

Table 1. Side-by-side comparison of ground truth k and estimated k

Dataset	Ground truth k	Estimated k of algorithm	Deviation from ground truth	Total run time (sec)	Time for finding optimal grid size (sec)	Time for centroid evaluation and value of k (sec)
a3	50	47	3	11.50498	11.47806	0.00409
s3	15	15	0	11.46858	11.44938	0.00263
s4	15	16	1	10.88031	10.86314	0.00296
<i>Birch1</i>	<i>100</i>	<i>412</i>	<i>312</i>	<i>222.207703</i>	<i>221.70673</i>	<i>0.07571</i>

4 Discussion

4.1 Assumptions

We make the following assumptions for the dataset. First, the dataset displays Gaussian distribution or near normal distribution. Second, clusters have a low degree of overlap. In other words, there is a certain separation measure between clusters. After all, human vision could also start to have difficulty identifying clusters with a high degree of overlap. To quantify this, in our study we used a standard deviation of no more than 42% distance between centroids, at a tolerance level of 0.5%.

4.2 Feature

As a pre-processing step that contributes to clustering, our algorithm offers the following advantages:

Robustness to Varying Densities. Our algorithm performed well on dataset S-3 where each cluster has varying densities, identifying all 15 ground truth clusters and finding centroids very close to the ground truth.

Robustness to Clusters of Different Shapes and Sizes. Our algorithm proved to be robust in cases where clusters are of different shapes. In the case of dataset S-4, one-third of the ground truth clusters are elongated as opposed to circular. Our algorithm identified all 15 ground truth clusters, although it found an extra cluster having been distracted by some outliers.

Guaranteed Termination of Algorithm in Polynomial Time. Our method of grouping the data into subregions and mapping the data in each region means that all data points are considered in the search for local maxima. The search will stop once the dataset is covered and the cut-off tolerance is reached. The guaranteed termination can help to limit the runtime and to ensure the algorithm reaches optimal accuracy.

Also, the complexity of the algorithm is $O(N \ln(M) + M^2 \ln(M))$, which is bounded above by $O(X^3)$, where X is the larger one between N and M . It is also worth noting that it is a very conservative estimation of the complexity as the worse cases for finding the optimal grid size and finding local maxima are considered separately without taking

into account the trade-off between the two processes. If a dataset requires a smaller grid size, it will need less steps in finding M but gives a larger value for M . This means the actual worst complexity of the algorithm will always be less than $O(N \ln(M) + M^2 \ln(M))$.

Other factors like the density of the data points, amount of noise and the shape of the clusters will also affect the actual run time, but their influences cannot be properly quantified as they will be different for individual dataset depending on the nature of each of them.

4.3 Critique

The main strength of our algorithm is perhaps its intuitive simplicity. On the flip side, complex cases and edge cases might not be the forte here.

Our initial assumptions of Gaussian distribution and non-overlapping clusters are also limitations. When the dataset presents some unusual situations violating our assumptions, our algorithm would underperform. For instance, where a single cluster contains more than two local maxima and displays varying densities at the same time, our algorithm cannot handle this complexity. If the cluster sizes vary radically, our algorithm is also not ideal.

When dealing with more complex cluster shapes, our grid can misalign with the cluster shape, resulting in less accurate centroid estimation. While this may be an acceptable margin of error for a pre-processing step, this is certainly an area for improvement.

4.4 Possible Improvements/Further Research

Clarity of Clusters. After testing our algorithm on the Clustering Datasets, several areas for improvement have been revealed. First of all, a score may be needed to indicate how clear-cut the clusters are. This is again applying a concept in human cognition to enhance the algorithm – a human can intuitively tell whether an image is clear or not. This would become more important when we try to apply the methods used here to higher dimension data.

Underclustering. A related issue is underclustering, which we saw in some cases such as seen in the Birch 1 dataset. This could be due to radical differences in cluster size, density or shapes within a given dataset. Even so, it would be beneficial to optimize the algorithm for dealing with outliers or noise.

Optimal Grid Size. Another area for further investigation is how to determine the optimal grid size. Our current method sets a tolerance level. It is worth exploring how we can use a method that is less arbitrary or less based on trial-and-error to determine the appropriate grid size.

Improving Computational Efficiency, Optimizing Exponential Step. The exponential step, currently set at 0.975, is thorough for our purposes but leads to numerous iterations. However, a larger exponential step may miss the optimal grid size.

We would need to find a way to decrease the number of iterations, especially at higher dimensions. This is very important as the runtime for finding the optimal grid size takes up over 95% of the total runtime for the current algorithm.

4.5 Value

As unsupervised learning is often highly dependent on input parameters from the user and the dataset's ground truth is not available, our algorithm is an attempt to apply human cognition to help. To a certain extent, our algorithm has shown to be informative, taking out the guesswork of the appropriate number of clusters to test for a clustering algorithm.

Given its robustness to differing cluster densities and shapes, although only in two dimensions, there may be potential in extending our method to higher dimensions. In addition, by indicating a suitable grid size, our algorithm can also be used to provide a good standard deviation value for kernel density clustering.

5 Conclusion

Grouping data into sub-regions provides valuable insights into the data distributions. By searching for local maxima in data densities, it can even give an estimation of number of clusters and centroid estimation to a satisfactory level of accuracy of maximum deviation of 3 data points if the data follows a Gaussian-like distribution and with SD lower than 42.4% of distances to the nearest neighboring cluster. Further research is needed to provide better results and more insights into human cognition in all levels will bring great value to the development of machine learning algorithms and artificial intelligence.

References

1. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. *ACM SIGKDD Explor. Newsl.* **4-1**, 65–75 (2002)
2. Wong, K.-C.: A short survey on data clustering algorithms (2015). arxiv.org/pdf/1511.09123.pdf
3. Jain, A.K., et al.: Data clustering: a review. *ACM Comput. Surv. (CSUR)* **31**(3), 264–323 (1999)
4. Yadav, J., Sharma, M.: A review of K-mean algorithm. *Int. J. Eng. Trends Technol. (IJETT)* **4**(7), 2972–2976 (2013)
5. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: Clustering validity checking methods: part I and part II. *ACM SIGMOD*, **31**(2), 40–45 and **31**(3), 19–27 (2002a, 2002b)
6. Kriegel, H.-P., Kröger, P., Sander, J., Zimek, A.: Density-based clustering. *WIREs Data Min. Knowl. Discov.* **1**(3), 231–240 (2011)
7. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008)
8. Kohonen, T., Honkela, T.: Kohonen network. *Scholarpedia* **2**(1), 1568 (2007)

9. Bação, F., Lobo, V., Painho, M.: Self-organizing maps as substitutes for k-means clustering. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3516, pp. 476–483. Springer, Heidelberg (2005). https://doi.org/10.1007/11428862_65
10. Wong, K.-C., Peng, C., Li, Y., Chan, T.-M.: Herd clustering: a synergistic data clustering approach using collective intelligence. *Appl. Soft Comput.* **23**, 61–75 (2014)
11. Tan, F., Slotin, J.-J.: A quorum sensing inspired algorithm for dynamic clustering (2015). arxiv.org/pdf/1303.3934.pdf
12. Santosh, D.: Tracking multiple moving objects using gaussian mixture model. *Int. J. Soft Comput. Eng.* **3**(2), 114–119 (2013)
13. Han, S., Humphreys, G.W., Chen, L.: Uniform connectedness and classical gestalt principles of perceptual grouping. *Percept. Psychophys.* **61**, 661–674 (1999)