SoftwareX 17 (2022) 100944

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

DCEM: An R package for clustering big data via data-centric modification of Expectation Maximization

Parichit Sharma [a,*], Hasan Kurban [a,b], Mehmet Dalkilic [a]

[a] Computer Science Department, Luddy School of Informatics, Computing, and Engineering, Indiana University, 700 N. Woodlawn Ave., Bloomington, 47408, IN, USA

[b] Computer Engineering Department, Siirt University, 56100 Siirt, Turkey

ARTICLE INFO

Article history:
Received 20 September 2021
Received in revised form 11 December 2021
Accepted 13 December 2021

Keywords:
Data centric machine learning
Big data
Unsupervised clustering
Expectation Maximization
Open source software

ABSTRACT

Clustering is intractable, so techniques exist to give a best approximation. Expectation Maximization (EM), initially used to impute missing data, is among the most popular. Parameters of a fixed number of probability distributions (PDF) together with the probability of a datum belonging to each PDF are iteratively computed. EM does not scale with data size, and this has hampered its current use. Using a data-centric approach, we insert hierarchical structures within the algorithm to separate high expressive data (HE) from low expressive data (LE): the former greatly affects the objective function at some iteration *i*, while LE does not. By alternating using either HE or HE+LE, we significantly reduce run-time for EM. We call this new, data-centric EM, EM*. We have designed and developed an R package called **DCEM** (**D**ata **C**lustering with **E**xpectation **M**aximization) to emphasize that data is driving the algorithm. DCEM is superior to EM as we vary size, dimensions, and separability, independent of the scientific domain. DCEM is modular and can be used as either a stand-alone program or a pluggable component. DCEM includes our implementation of the original EM as well. To the best of our knowledge, there is no open source software that specifically focuses on improving EM clustering without explicit parallelization, modified seeding, or data reduction. DCEM is freely accessible on CRAN (Comprehensive R Archive Network).

Code metadata

Current code version	v2.0.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-21-00177
Legal Code License	GPL-3
Code versioning system used	GIT
Software code languages, tools, and services used	R >= 3.5
Compilation requirements, operating environments & dependencies	R >= 3.5, matrixcalc, mvtnorm, MASS, Rcpp
If available Link to developer documentation/manual	https://cran.r-project.org/web/packages/DCEM/DCEM.pdf
Support email for questions	parishar@iu.edu

1. Introduction

The well-known physicist Feynman observed more than 50 years ago that over time, the ratio of compute time to the size of data would approach zero [1]. What is surprising is the rate at which we have been approaching zero! Data-centric AI [2,3] is a response to this growing challenge by focusing on data, its preparation and management, and how, when iterating to an optimum, not only does the value of data change, but the value can be leveraged in more efficient algorithms. The first data-centric AI workshop has been established this year [4].

While simple to understand, clustering is difficult because it lacks theoretical foundations and its search space has a faster than exponential growth (brute force, even for very small sizes is impossible). The direct consequence is that clustering algorithms are approximations that fix *a priori*, for instance, the number of clusters. Among the most popular approaches is Expectation-Maximization (EM) initially developed for imputing missing values. EM presumes a fixed number of clusters (or centroids) that

* Corresponding author.
E-mail address: parishar@iu.edu (Parichit Sharma).

https://doi.org/10.1016/j.softx.2021.100944

are, in fact, probability distributions (PDFs). The technique iteratively finds the expectation of parameters of the PDFs and the maximization of probability a datum belonging to each PDF. Convergence is on the log probability of these values. Unfortunately, in the era of big data (volume, variety, velocity), EM does not scale. We observe that no formal definition of "big data" exists. Rather, its a consensus that now data exists in multiple ways that pose challenges to traditional AI algorithms many of which are several decades old. As an example from our work [5], a graph built on functional genomes in *D. melanogaster* has 25 K nodes. There are $O(9.8 \times 10^{21})$ paths of length five which is biologically small but computationally enormous.

To modernize EM, we leverage a data-centric approach: data should drive and affect the algorithm directly as iteration takes place. Observe that in some iterative algorithms, the data can be separated based on how it affects the objective function [6]. As EM iterates, we ignore data that does not affect the objective function (low expressive data) while using data that does (high expressive data). We insert data structures that capture regions of LE+HE, alternating between using HE or LE+HE corresponding to expectation and maximization. Both the re-building and insertion of data are very efficient.

We have chosen R [7], since it is open source, has a vibrant and active community, and is among the most widely used languages/platforms for data analytics.

1.1. A deeper look into DCEM

EM-T (we use 'T' to denote the traditional EM) is a parametric learning technique that initially was used to estimate missing data [8], but today is a standard tool, multifaceted both in its use and in diverse areas [9–11]. One particularly popular use of EM-T is for *soft* clustering in which each datum is associated with *all* centroids. The use of EM-T in clustering presumes that each datum comes from a particular probability distribution–the simplest approach is to assume normal distribution. In order to assign data to the clusters, the parameter values of each probability distribution for each cluster are iteratively updated by maximizing the log-likelihood function. EM-T converges as the data points stop shifting among the clusters—in other words, the probabilities stabilize. In keeping with historical use, DCEM includes both original implementation of EM-T and our data-centric extension EM* (EM-star). It should be remarked that there are, in fact, many slightly differing versions of the EM-T algorithm. We have chosen [12] for this package. Additionally, we include an improved seeding method.

1.2. Related work

CRAN has several packages that address the problem of unsupervised learning and model-based clustering. While an exhaustive list can be accessed at the CRAN task view [13], Table 1 presents a feature based comparison of packages (that use EM-T or its variant): A comprehensive discussion of the available R packages and their merits is also provided in the supplementary information. Whereas most packages provide tailored solutions to the problem of unsupervised clustering, not much significant work has been devoted to making EM-T work with *actual big data* (as also discussed in [14]).

1.3. Background & motivation

Big data, and its now typical pairing with the popular area of data science, has exposed an obvious weakness of iterative algorithms—while the run-time of visiting the data is $O(n)$, as n routinely runs into sizes of 100 s of gigabytes and even several terabytes (and, in the near future petabytes) and dimensions regularly spanning 100 s into 1000s of features, the operation of visiting each and every datum overwhelms many traditional iterative algorithms. To address this gap, we designed and implemented DCEM to specifically work effectively on big data—'high dimensions, large sizes, and disparate domains. DCEM (EM∗) not only dramatically improves the run-time on big data, but also achieves competitive accuracy rates in almost all cases [14].

Table 1
MB is Model Based Clustering. A listing of related CRAN packages and their functions as of (09/03/2021).

Pkg.	Vrs.	Clust./class	Density est.	MB clust.	Parallel
mclust	5.4.6	Y/Y	Y	Y	N
mixture	1.5	Y/Y	N	Y	N
Rmixmod	2.1.5	Y/Y	Y	Y	N
flexmix	2.3–15	Y/N	N	Y	N
EMCluster	0.2–12	Y/Y	N	Y	N
mixtools	1.2.0	Y/N	Y	N	N
ClusterR	1.2.2	Y/Y	N	N	Y
DCEM (EM*)	2.0.2	Y/Y	N	N	N

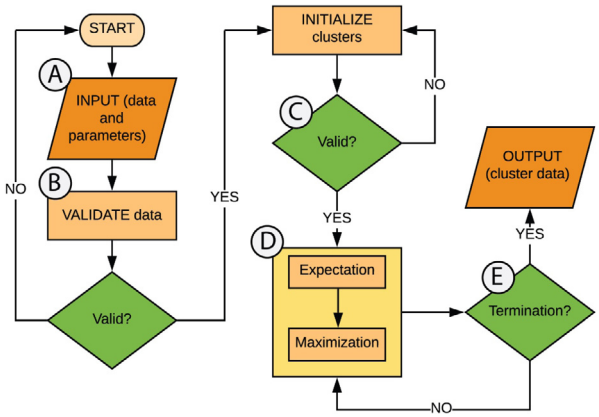

Fig. 1. Execution process of EM∗ where ϵ is convergence threshold. (A) The package requires data frames. (B) For example, invalid parameters specified, missing/invalid values in data. (C) For example, empty blocks, fewer modalities in data than specified by user. (D) Either EM-T or EM∗. (E) EM∗: Check if $\epsilon \leq 0.01$ (99% of the leaves are same across the heaps) or maximum iterations reached. EM-T: Check if $\epsilon \leq$ convergence threshold or maximum iterations reached.

2. Software description & workflow

DCEM is organized into modular scripts for performing tasks such as reading data from files, pre-processing and cleaning the data, executing the expectation and maximization procedures, and formatting the output data. An abstract view of the package execution workflow is given in (Fig. 1).

2.1. Package features

DCEM provides an intuitive interface for both EM∗ and EM-T algorithms that can be easily customized via function call arguments. It also supports advanced initialization schemes [15, 16] for expediting the convergence process. To further enhance performance, code vectorization is used extensively for matrix calculations and, routines for heap creation and initialization are implemented in Rcpp [17]–features lacking in the original Python code. To get started with the package, users can directly access help and documentation via R console. Additionally, detailed examples with code walk through, various use cases and comprehensive explanation of the package API is also found in the user manual [18] and supplementary quick start document.

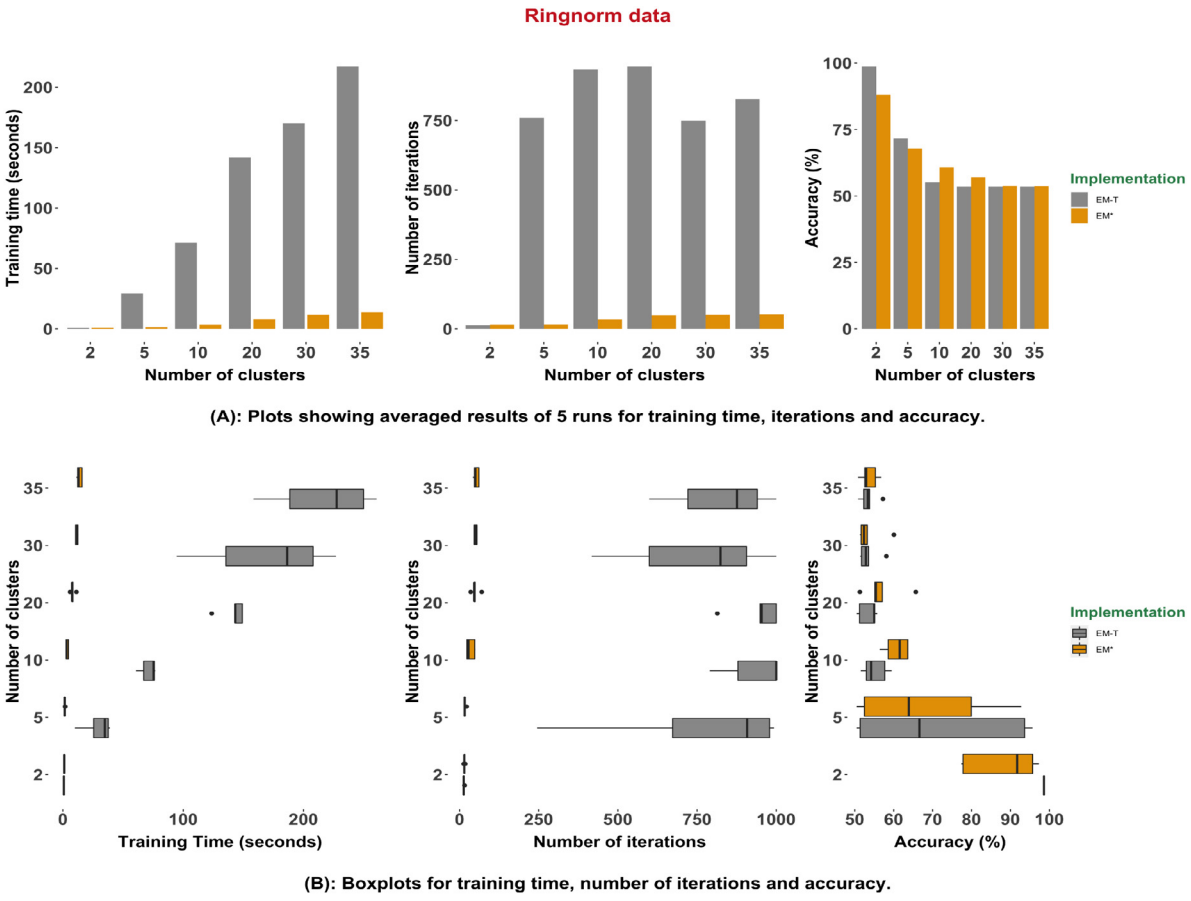

Fig. 2. In case of execution time and iterations, EM∗ is significantly better than EM-T for all values of k except for $k = 2$ (where EM-T is marginally better). Accuracy plots show that EM-T is slightly better for $k = \langle 2, 5\rangle$ while EM∗ is better for $k = \langle 10, 20\rangle$, and both algorithms perform similarly for $k > 20$.

3. Illustrative examples

A comparison of EM∗ and EM-T on a variety of real and synthetic data sets using the older Python code is given in [14]. Here, we illustrate the utility of DCEM by benchmarking on the ringnorm data [19] and extend the comparison on large data to selective R packages (additional benchmarks are given in the supplementary information). The results (Fig. 2) demonstrate that EM∗ outperforms EM-T in terms of execution time and number of iterations while preserving or even improving accuracy.

As mentioned in Table 1, CRAN has a multitude of software that use, in some way, EM-T, but not all of them provide EM-T as the *main* routine or focus on extending EM-T to work on big data; therefore, we selected those packages for comparison that: (1) do not impose restrictions on specific properties of data (shape, volume or orientation of the distribution) by using predefined models, (2) provide an integrated (combining both E and M steps) API to directly invoke the EM algorithm, (3) support random initialization and, (4) implement the E and M steps in the R language. We are left with examining **EMCluster** [20] and **mixtools** [21]. The experimental results are shown in Fig. 3 and Fig. 4. To obtain the results in reasonable time, we set a time limit of 2 h on all the algorithms to prevent them from running for very long time and the runs are not repeated.

The user may reproduce the selective results by consulting the supplementary information.

4. Impact & conclusion

Clustering (or more formally, partitioning) is the first step in understanding data. While the most primitive kind of analysis, clustering does not possess any unified theoretical foundations; further, it is intractable. The most obvious initial attempt to understand data is ironically the most difficult. This has constrained feature number, centroid number, and data sizes. DCEM is built on a data-centric principle of *data expressiveness* which captures how the value of data changes as it is revisited in iterative optimization problems. We identify a gamut of data expressiveness from low (LE) to high (HE) that affects the objective function less or more, respectively. Heaps, we discovered, possess correlated expressive regions we call *strong* (LE data) and *weak* (HE data). By sometimes ignoring the strong region, we iterate over significantly less data [23].

Since the release (September, 2018), DCEM has been downloaded 18,000 times (Fig. 1, supplementary information) which also indicates its popularity in broader open source community. Our belief is that the number of downloads is related to the success of scientists examining significantly larger data sizes. We demonstrated that compared to existing software, significant improvements in run-time and number of iterations are achieved by DCEM as the data complexity (size, dimensions, and clusters) exceed a certain limit.

In summary, DCEM is a domain independent program and, can be used both as (1) a stand-alone program for clustering; (2) a plug-and-play component to be incorporated into existing software—more so as a supplement to the existing EM-T implementations to expedite parameter estimation.

DCEM provides an open source implementation for both EM-T (widely used in several areas [24–26]) and EM∗ (extension of EM-T), which significantly improves the performance on big data. DCEM can expedite the scientific discovery process particularly

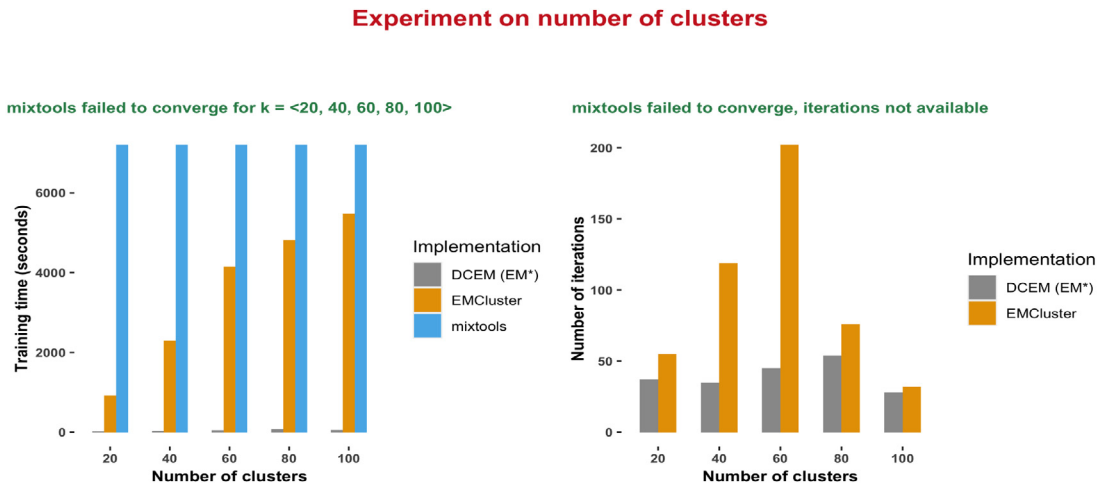

Fig. 3. Comparison on large number of clusters: In terms of training time, DCEM is significantly faster than EMCluster and mixtools. DCEM requires fewer iterations than EMCluster, iterations for mixtools are not reported as it could not converge in the threshold of 2 h. Plot created via ggplot2 [22].

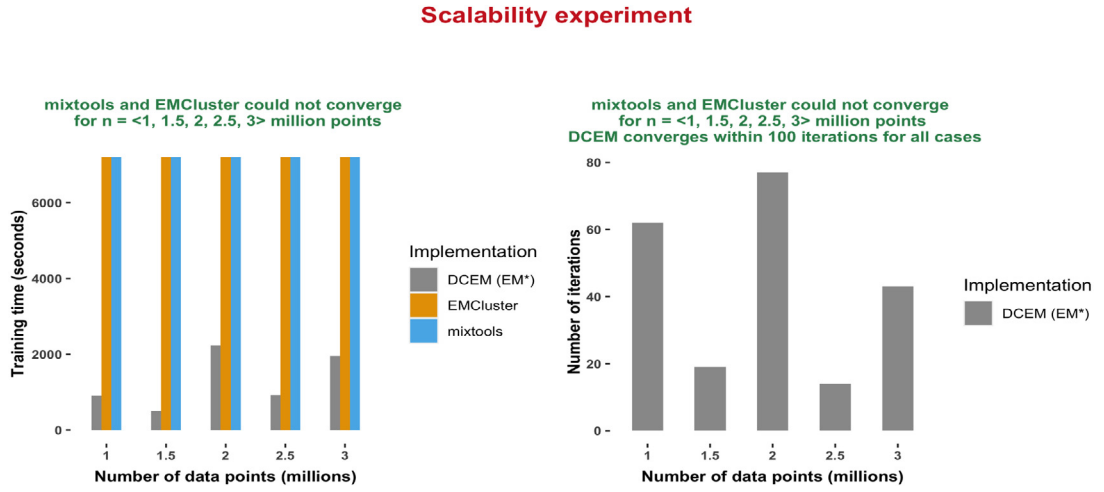

Fig. 4. Comparison on large data: DCEM always converges (maximum execution time of ∼33 min for 2 million points), whereas EMCluster and mixtools failed to converge within the time limit of 2 h. Iterations are not reported for mixtools and EMCluster because they failed to converge for all values of n. Plot created with ggplot2 [22].

for unsupervised clustering, as also illustrated via multitude of experiments in [2,27].

CRediT authorship contribution statement

Parichit Sharma: Methodology, Software, Validation, Investigation, Writing – original draft, Writing – review & editing, Visualization, Project administration. **Hasan Kurban:** Methodology, Investigation, Validation, Writing – review & editing, Visualization, Supervision. **Mehmet Dalkilic:** Conceptualization, Writing – original draft, Writing – review & editing, Resources, Supervision.

Declaration of competing interest

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

Acknowledgment

The authors appreciate the technical support extended by Rob Henderson (Computer Science Department) for his help in deploying the system infrastructure for this work.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.softx.2021.100944.

References

[1] Feynman R. Lectures on computation (Frontiers in physics). 1st Ed.. CRC Press; 2000.
[2] Kurban H. A novel approach to optimization of iterative machine learning algorithms over heap structures (Ph.D. thesis), Indiana University; 2017.
[3] Kurban H, Sharma P, Dalkilic M. Data Expressiveness and Its Use in Data-centric AI, Data Centric AI Workshop, 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia, https://datacentricai.org/papers/145_CameraReady_Poster_DCAI_DataExpressiveness_2021.pdf.
[4] Data centric artificial intelligence. NeurIPS; 2021, URL http://datacentricai.org/.
[5] Costello JC, Dalkilic MM, Beason SM, Gehlhausen JR, Patwardhan R, Middha S, Eads BD, Andrews JR. Gene networks in Drosophila melanogaster: integrating experimental data to predict gene function. Genome Biol 2009;10(9):1–29.
[6] Wolpert DH. Ubiquity symposium: Evolutionary computation and the processes of life: What the no free lunch theorems really mean: How to improve search algorithms. Ubiquity 2013;2013(December):1–15.

[7] RCore Team. R: A language and environment for statistical computing. Vienna, Austria: R Foundation for Statistical Computing; 2021, URL https://www.R-project.org/.
[8] Dempster A, Laird N, Rubin D. Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc Ser B Stat Methodol 1977;39(1):1–22.
[9] Yuille A, Stolorz P, Utans J. Statistical physics, mixtures of distributions, and the EM algorithm. Neural Comput 1994;6(2):334–40.
[10] Xu L, Jordan M. On convergence properties of the EM algorithm for Gaussian mixtures. Neural Comput 1996;8(1):129–51.
[11] Roweis S, Ghahramani Z. A unifying review of linear Gaussian models. Neural Comput 1999;11(2):305–45.
[12] Zaki MJ, Meira Jr. W, Meira W. Data mining and analysis: Fundamental concepts and algorithms. Cambridge University Press; 2014.
[13] Leisch F GB. CRAN task view: Cluster analysis & finite mixture models. 2012, URL http://CRAN.R-project.org/view=Cluster.
[14] Kurban H, Jenne M, Dalkilic M. Using data to build a better EM: EM* for big data. Int J Data Sci Anal 2017;4(2):83–97.
[15] Bahmani B, Moseley B, Vattani A, Kumar R, Vassilvitskii S. Scalable k-means++. Proc Very Large Data Bases Endow (PVLDB) 2012;5(7):622–33.
[16] Arthur D, Vassilvitskii S. K-means++: The advantages of careful seeding. In: Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. SODA '07, USA: Society for Industrial and Applied Mathematics; 2007, p. 1027–35.
[17] Eddelbuettel D, Balamuta JJ. Extending extitr with extitC++: A brief introduction to Rcpp. PeerJ Preprints 2017;5:e3188v1. http://dx.doi.org/10.7287/peerj.preprints.3188v1.
[18] Sharma P. Clustering big data using expectation maximization star (EM*) algorithm. CRAN; 2019, URL https://cran.r-project.org/web/packages/DCEM/DCEM.pdf.
[19] Breiman L. Bias, variance and arcing classifiers. 1996, URL https://www.stat.berkeley.edu/~breiman/arcall96.pdf.
[20] W.C. C, R. M. EMCluster: EM algorithm for model-based clustering of finite mixture Gaussian distribution. 2015, URL http://cran.r-project.org/package=EMCluster.
[21] Benaglia T, Chauveau D, Hunter DR, Young D. Mixtools: An R Package for analyzing finite mixture models. J Stat Softw 2009;32(6):1–29, URL http://www.jstatsoft.org/v32/i06/.
[22] Wickham H. Ggplot2: Elegant graphics for data analysis. Springer-Verlag New York; 2016, URL https://ggplot2.tidyverse.org.
[23] Kurban H, Dalkilic M. A novel approach to optimization of iterative machine learning algorithms: Over heap structures. In: 2017 institute of electrical and electronics engineers, international conference on big data (Big data). 2017, p. 102–9. http://dx.doi.org/10.1109/BigData.2017.8257917.
[24] Do C, Batzoglou S. What is the expectation maximization algorithm? Nature Biotechnol 2008;26(8):897–9.
[25] Jung Y, Kang M, Heo J. Clustering performance comparison using K-means and expectation maximization algorithms. Biotechnol Biotechnol Equip 2014;28(sup1):S44–8.
[26] McLachlan G, Krishnan T. The EM algorithm and extensions. vol. 382, John Wiley & Sons; 2007.
[27] Kurban H, Kockan C, Jenne M, Dalkilic MM. Case Study: Clustering Big Stellar Data with EM. In: Proceedings of the fourth IEEE/ACM international conference on big data computing, applications and technologies; 2017. P. 271–2.