Information Sciences 217 (2012) 31–38

Contents lists available at SciVerse ScienceDirect

Information Sciences

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

Clustering by analytic functions

Mikko I. Malinen *, Pasi Fränti

Speech and Image Processing Unit, School of Computing, University of Eastern Finland, P.O. Box 111, FIN-80101 Joensuu, Finland

ARTICLE INFO

Article history:
Received 27 January 2010
Received in revised form 12 April 2012
Accepted 10 June 2012
Available online 26 June 2012

Keywords:
Clustering
Analytic function
Mean squared error
k-Means

ABSTRACT

Data clustering is a combinatorial optimization problem. This article shows that clustering is also an optimization problem for an analytic function. The mean squared error, or in this case, the squared error can expressed as an analytic function. With an analytic function we benefit from the existence of standard optimization methods: the gradient of this function is calculated and the descent method is used to minimize the function.

1. Introduction

Euclidean sum-of-squares clustering is an NP-hard problem [2], where we group n data points into k clusters. Each cluster has a centre (centroid) which is the mean of the cluster and one tries to minimize the mean squared distance (mean squared error, MSE) of the data points from the nearest centroid. When the number of clusters k is constant, this problem becomes polynomial in time and can be solved in $O(n^{kd+1})$ time [14]. Although polynomial, this problem is slow to solve optimally. In practice, suboptimal algorithms are used. The method of k-means clustering [17] is fast and simple, although its worst-case running time is superpolynomial with a lower bound of $2^{\Omega(\sqrt{n})}$ for the number of iterations [3].

Given a set of observations $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)$, where each observation is a d-dimensional real vector, then k-means clustering aims to partition the n observations into k sets $(k < n)$ $\mathbf{S} = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sums of squares:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2, \tag{1}$$

where $\boldsymbol{\mu}_i$ is the mean of S_i. Given an initial set of k means $\mathbf{m}_1^{(1)}, \ldots, \mathbf{m}_k^{(1)}$, which may be specified randomly from the set of data points or by some heuristic [19,22,4], the k-means algorithm proceeds by alternating between two steps: [16].

Assignment step: Assign each observation to the cluster with the closest mean (i.e. partition the observations according to the Voronoi diagram generated by the means).

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leqslant \|\mathbf{x}_j - \mathbf{m}_{i^*}^{(t)}\| \quad \forall \quad i^* = 1, \ldots, k \right\}. \tag{2}$$

Update step: Calculate the new means as the centroids of the observations in each cluster:

* Corresponding author.
E-mail addresses: mmali@cs.uef.fi (M.I. Malinen), franti@cs.uef.fi (P. Fränti).
URLs: http://cs.uef.fi/~mmali (M.I. Malinen), http://cs.uef.fi/~franti (P. Fränti).

0020-0255/$ - see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2012.06.018

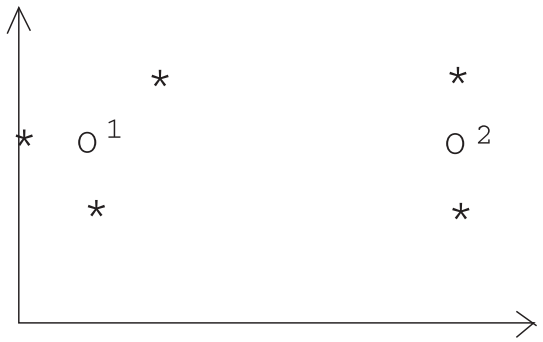

Fig. 1. A set of two clusters $i = 1, 2$ with five data points (0,3), (1,2), (2,4), (8,2), (8,4) in two dimensions (features) $j = 1, 2$. The feature j of data point k is represented as x_{kj}.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{\left|S_i^{(t)}\right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j. \tag{3}$$

The algorithm has converged when the assignments no longer change.

The advantage of k-means is that it finds a locally optimized solution for any given initial solution by repeating this simple two-step procedure. However, k-means cannot solve global problems in the clustering structure, and thus, it will work perfectly only if the global cluster structure is already optimized. By optimized global clustering structure we mean centroid locations from which optimal locations can be found by k-means. This is the main reason why slower agglomerative clustering is sometimes used [10,13,12], or other more complex k-means variants [11,18,4,15] are applied. Gaussian mixture models can be used (Expectation–Maximization algorithm) [8,25] and cut-based methods have been found to give competitive results [9]. To get a glimpse of the recent research in clustering, see [1,24,26], which deal with particle swarm optimization, ant-based clustering and minimum spanning tree based split-and-merge algorithm.

The method presented in this paper corresponds to k-means and is based on representing the squared error (SE) as an analytic function. The MSE or SE value can be calculated when the data points and centroid locations are known. The process involves finding the nearest centroid for each data point. An example dataset is shown in Fig. 1. We write c_{ij} for the centroid of cluster i, feature j. The squared error function can be written as

$$f(\bar{c}) = \sum_u \min_i \left\{ \sum_j (c_{ij} - x_{uj})^2 \right\}. \tag{4}$$

The min operation forces one to choose the nearest centroid for each data point. This function is not analytic because of the min operations. A question is whether we can express $f(\bar{c})$ as an analytic function which then could be given as input to a gradient-based optimization method. The answer is given in the following section.

2. Analytic clustering

2.1. Formulation of the method

We write the p-norm as

$$\|\bar{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}. \tag{5}$$

The maximum value of x_i's can be expressed as

$$\max(|x_i|) = \lim_{p\to\infty} \|\bar{x}\|_p = \lim_{p\to\infty} \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{6}$$

Since we are interested in the *minimum* value, we take the inverses $\frac{1}{x_i}$ and find their maximum. Then another inverse is taken to obtain the minimum of the x_i:

$$\min(|x_i|) = \lim_{p\to\infty} \left(\sum_{i=1}^{n} \frac{1}{|x_i|^p} \right)^{-1/p} \tag{7}$$

2.2. Estimation of infinite power

Although calculations of the infinity norm without comparison operations are not possible, we can estimate the exact value by setting p to a high value. The estimation error is

$$\epsilon = \left(\sum_{i=1}^{n} \frac{1}{|x_i|^p}\right)^{-1/p} - \lim_{p_2 \to \infty} \left(\sum_{i=1}^{n} \frac{1}{|x_i|^{p_2}}\right)^{-1/p_2} \tag{8}$$

The estimation can be made up to any accuracy, the estimation error being

$$|\epsilon| > 0.$$

To see how close we can come in practice, a mathematical software package run was made:

$$1/\text{nthroot}((1/x1)\hat{}p + (1/x2)\hat{}p, p).$$

For example, with the values $x1,\ x2 = 500,\ p = 100$ we got the result 496.54. When the values of $x1$ and $x2$ are far from each other, we get an accurate estimate, but when the numbers are close to each other, an approximation error is present. In Table 1, the inaccuracy of the estimate is shown for different values of p and x_i. In this table, the estimate with two equal values $x_1 = x_2$ is calculated. In Fig. 2, the inaccuracy is calculated as a function of p. In this example, p cannot be increased much more, although it would give a more accurate answer. In Fig. 3, we see how large values of p can be used in maximum value calculations with this package. Moreover, in Fig. 4, we see how accurate the estimates can be using these maximum powers. On the basis of these results, we recommend scaling the values of x_i to the range $[0.5, 2]$ to achieve the best accuracy. Typically, dataset values are integers and range in magnitude from 0 to 500 or floats and range in magnitude from 0 to 1.

2.3. Analytic formulation of SE

Combining (4) and (7) yields

$$f(\bar{c}) = \sum_u \left[\lim_{p \to \infty} \left(\left(\sum_i \frac{1}{\left| \sum_j (c_{ij} - x_{uj})^2 \right|^p} \right)^{-1/p} \right) \right]. \tag{9}$$

Proceeding from (9) by removing lim, we can now write $\hat{f}(\bar{c})$ as an estimator for $f(\bar{c})$:

$$\hat{f}(\bar{c}) = \sum_u \left[\left(\sum_i \left(\sum_j (c_{ij} - x_{uj})^2 \right)^{-p} \right)^{-\frac{1}{p}} \right]. \tag{10}$$

This is an analytic estimator, although the exact $f(\bar{c})$ cannot be written as an analytic function when the data points lie in the middle of cluster centroids in a certain way.

Partial derivatives and the gradient can also be calculated. The formula for partial derivatives is calculated using the chain rule:

$$\frac{\partial \hat{f}(\bar{c})}{\partial c_{st}} = \sum_u \left[-\frac{1}{p} \cdot \left(\sum_i \left(\sum_j (c_{ij} - x_{uj})^2 \right)^{-p} \right)^{-\frac{p+1}{p}} \cdot \sum_i \left(-p \cdot \left(\sum_j (c_{ij} - x_{uj})^2 \right)^{-(p+1)} \right) \cdot 2 \cdot (c_{st} - x_{ut}) \right]. \tag{11}$$

Table 1
Inaccuracy of the estimate of the maximum value of ((6)) as p and x_i, $(i = 1,\ 2)$ change.

p	$x_i = 1$ (%)	$x_i = 10$ (%)	$x_i = 100$ (%)	$x_i = 500$ (%)
0	100	100	100	100
10	7	7	7	7
20	3	3	3	3
30	2	2	2	2
40	2	2	2	2
50	1.4	1.4	1.4	1.4
60	1.1	1.1	1.1	1.1
70	1.0	1.0	1.0	1.0
80	0.9	0.9	0.9	0.9
90	0.8	0.8	0.8	0.8
100	0.7	0.7	0.7	0.7
110	0.6	0.6	0.6	0.6
120	0.6	0.6	0.6	N/A

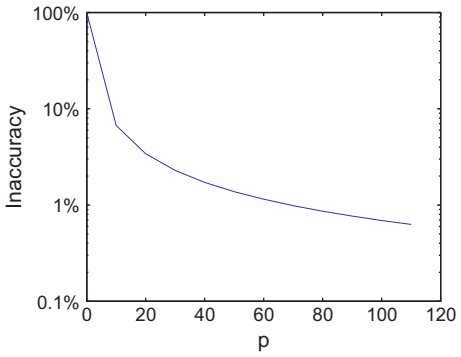

Fig. 2. Inaccuracy of estimate of the maximum value of ((6)) as a function of p ($x_i = 1$ to $x_i = 500,\ i = 1,\ 2$).

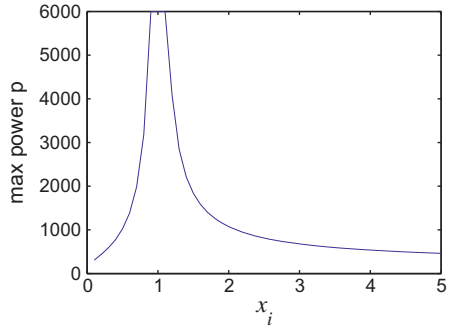

Fig. 3. Maximum power that can be calculated by a mathematical software package with different values of x_i.

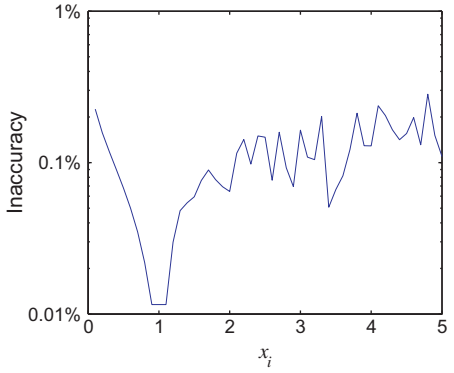

Fig. 4. Inaccuracy as a function of x_i, $i = 1,\ 2$, and when p is maximal.

2.4. Time complexity

For analysing the time complexity of calculating $\hat{f}(\bar{c})$, which is presented in ((10)), we know that $(\cdot)^{-p} = \frac{1}{(\cdot)^p}$ involves p divisions and that one division requires constant time in computer, and $(\cdot)^{\frac{1}{p}}$ takes $O(\log p)$ [7]. Using these, we can calculate

$$\begin{aligned} T(\hat{f}(\bar{c})) &= \left(d \cdot (\mathrm{Mult} + \mathrm{Add}) \cdot k \cdot (T(\wedge - p) + \mathrm{Add}) + T\left(\wedge - \frac{1}{p}\right)\right) \cdot n = O(d \cdot \mathrm{Mult} \cdot k \cdot T(\wedge - p) \cdot n) \\ &= O(d \cdot \mathrm{Mult} \cdot k \cdot p \cdot n) = O(n \cdot d \cdot k \cdot p). \end{aligned} \tag{12}$$

The time complexity of calculating $\hat{f}(\bar{c})$ grows linearly with the number of data points n, dimensionality d, number of centroids k, and power p.

To calculate the time complexity of the partial derivative (s), which are presented in ((11)), we divide this into three parts, A, B, C:

$$
\begin{aligned}
A &= \left(\sum_i \left(\sum_j (c_{ij} - x_{uj})^2\right)^{-p}\right)^{-\frac{p+1}{p}} \\
B &= \sum_i \left(-p \cdot \left(\sum_j (c_{ij} - x_{uj})^2\right)^{-(p+1)}\right) \\
C &= (c_{st} - x_{ut}).
\end{aligned} \tag{13}
$$

Knowing that $(\cdot)^{-\frac{p+1}{p}} = (\cdot)^{-1} \cdot (\cdot)^{-\frac{1}{p}}$, we can write

$$
\begin{aligned}
T(A) &= d \cdot (\text{Mult} + \text{Add}) \cdot k \cdot (T(\wedge - p) + \text{Add}) + T\left(\wedge - \frac{1}{p}\right), \\
T(B) &= O(d \cdot (\text{Mult} + \text{Add}) \cdot T(\wedge - (p+1)) \cdot k), \\
T(C) &= \text{Subtr},
\end{aligned} \tag{14}
$$

and

$$
\begin{aligned}
T(\text{partial derivative}) &= O(T(A) + T(B) + T(C)) \cdot n) = O(T(B) \cdot n) = O(d \cdot \text{Mult} \cdot T(\wedge(p+1)) \cdot k \cdot n) = O(d \cdot p \cdot k \cdot n) \\
&= O(n \cdot d \cdot k \cdot p).
\end{aligned} \tag{15}
$$

To calculate all partial derivatives, we have to calculate part C for each partial derivative. The parts A and B are the same for all derivatives. Since we calculate part C n times, and there are $k \cdot d$ partial derivatives, we get

$$
T(\text{all partial derivatives}) = O(ndkp + n \cdot T(C) \cdot k \cdot d) = O(ndkp + n \cdot k \cdot d \cdot \text{Subtr}) = O(ndkp). \tag{16}
$$

This is linear in time for n, d, k and p, and differs only by the factor p from one iteration time complexity of the k-means $O(k \cdot n \cdot d)$.

2.5. Analytic optimization of SE

Since we can calculate the values of $\hat{f}(\bar{c})$ and the gradient, we can find a (local) minimum of $\hat{f}(\bar{c})$ by the gradient descent method. In the gradient descent method the points converge iteratively to a minimum:

$$
\bar{c}_{i+1} = \bar{c}_i - \nabla \hat{f}(\bar{c}_i) \cdot l, \tag{17}
$$

where l is the step length. The value of l can be calculated at every iteration, starting from some l_{max} and halving it recursively until $\hat{f}(\bar{c}_{i+1}) < \hat{f}(\bar{c}_i)$.

Eq. (11) for the partial derivatives depends on p. For any $p \geqslant 0$, either a local or the global minimum of (10) is found. Setting p large enough, we get a satisfactory estimator $\hat{f}(\bar{c})$, although there is always some bias in this estimator and a p that is too small may lead to a different clustering result.

There is also an alternative way to minimize $\hat{f}(\bar{c})$. Minimizing $\hat{f}(\bar{c})$ to the global minimum could be done by solving all $\bar{c}$ from (18) and trying them, one at a time, in $\hat{f}(\bar{c})$, because at a minimum point (global or local) all components of the gradient must be zero:

$$
\sum_{ij} \left(\frac{\partial \hat{f}(\bar{c})}{\partial c_{ij}}\right)^2 = 0. \tag{18}
$$

This alternative way has only theoretical significance, since it is not known how to find all solutions of (18). There are at least $i_{max}!$ solutions to this equation, since from each solution (which surely exist), $i_{max}!$ solutions can be obtained by permuting the centroids.

The analytic clustering method presented here corresponds to the k-means algorithm [17]. It can be used to obtain a local minimum of the squared error function similarly to k-means, or to simulate the random swap algorithm [11] by changing one cluster centroid randomly. In the random swap algorithm, a centroid and a datapoint are chosen randomly, and a trial movement of this centroid to this datapoint is made. If the k-means with the new centroid provide better results than the earlier solution, the centroid remains swapped. Such trial swaps are then repeated for a fixed number of times.

Analytic clustering and k-means work in the same way, although their implementations differ. Their step length is different. The difference in the clustering result also originates from the approximation of the ∞-norm by the p-norm.

We have used an approximation to the infinity norm to find the nearest centroids for the datapoints, and used the sum-of-squares for the distance metric. The infinity norm, on the other hand, could be used to cluster with the infinity norm distance metric. Most partitioning clustering papers use the $p = 2$ (Euclidean norm) as the distance metric as we do, but some papers have experimented with different norms. For example, $p = 1$ gives the k-medians clustering, e.g. [23], and $p \to 0$ gives the categorical k-modes clustering. Papers on the k-midrange clustering (e.g. [6,20]) employ the infinity norm ($p = \infty$) in finding the range of a cluster. In [5] a $p = \infty$ formulation has been given for the more general fuzzy case. A description and comparison of different formulations has been given in [21]. With the infinity norm distance metric, the distance of a

data point from a centroid is the dominant feature of the difference vector between the data point and the centroid. Our contribution in this regard is that we can form an analytic estimator for the cost function even if the distance metric were the infinity norm. This would make the formula for $\hat{f}(\bar{c})$ and the formula for the partial derivatives a little bit more complicated but nevertheless possible as a future direction, and thus, it is omitted here.

3. Experiments

We test this new clustering method not by using the p-norm but using the min-function to calculate the distances to the nearest centroids and a line search instead of the gradient descent method. We use several small and mid-size datasets (see Fig. 5) and compare the results of the analytic clustering, the k-means clustering, the random swap clustering, and the analytic random swap clustering. The number of clusters is based on the known number of clusters in the datasets. The results are illustrated in Table 2 and show that analytic clustering and k-means clustering provide comparable results. In these experiments, the analytic random swap algorithm sometimes gives a better (lower) SE value than random swapping. We also calculated the Adjusted Rand index, a neutral measure of clustering performance beyond sum of squares, for ten runs of the analytic clustering and the k-means clustering as well as for the random swap variants of these. Runs are done for the s-sets. The means of the Rand indices are shown in Table 3. These results indicate that the clustering performance is very similar between the analytic and the traditional methods. The running time for the s-sets is reasonable (e.g., 4.6 s for analytic clus-

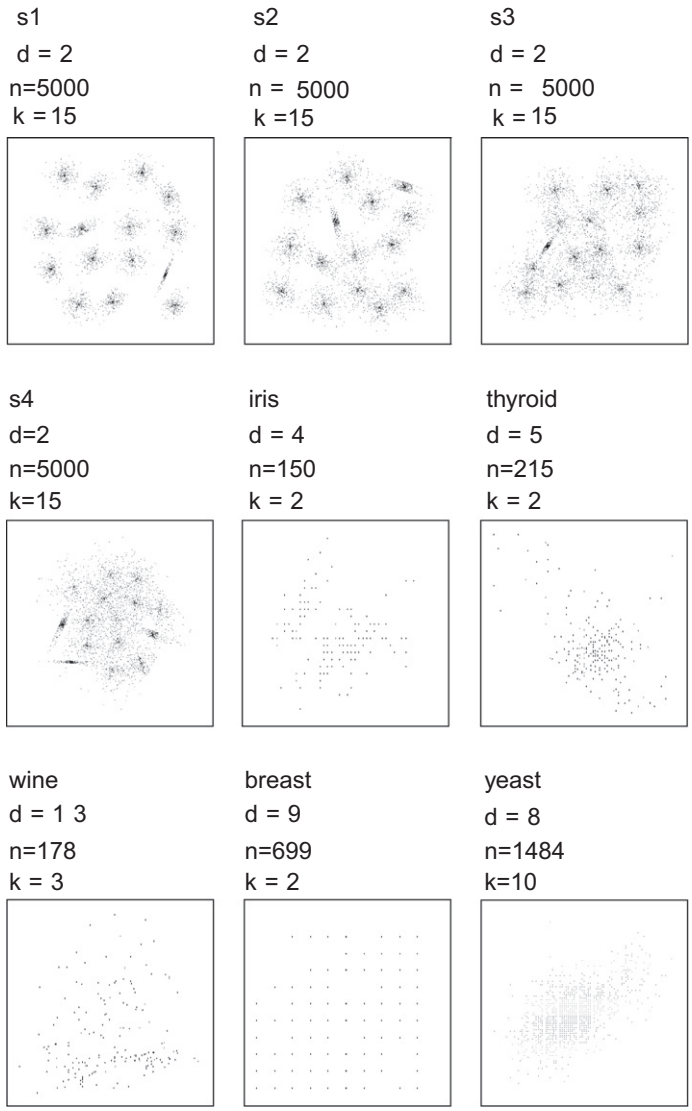

Fig. 5. Datasets $s1$, $s2$, $s3$, $s4$, iris, thyroid, wine, breast and yeast used in experiments. Two first dimensions are shown.

Table 2
Averages of SE values of 30 runs of analytic and traditional methods. The SE values are divided by 10^{13} or 10^6 (wine set) or 10^4 (breast set) or 1 (yeast set). Calculated using ((4)). Processing times in seconds for different datasets and methods.

Dataset	Squared error				Processing time			
	k-Means		Random swap		*k*-Means		Random swap	
	Anal.	Trad.	Anal.	Trad.	Anal.	Trad.	Anal.	Trad.
*s*1	1.93	1.91	1.37	1.39	4.73	0.04	52.46	0.36
*s*2	2.04	2.03	1.52	1.62	6.97	0.08	51.55	0.61
*s*3	1.89	1.91	1.76	1.78	4.59	0.06	59.03	0.58
*s*4	1.70	1.68	1.58	1.60	5.43	0.23	49.12	1.13
Iris	22.22	22.22	22.22	22.22	0.12	0.01	0.48	0.03
Thyroid	74.86	74.80	73.91	73.91	0.22	0.02	0.72	0.04
Wine	2.41	2.43	2.37	2.37	0.44	0.02	4.39	0.04
Breast	1.97	1.97	1.97	1.97	0.15	0.02	1.07	0.04
Yeast	48.87	48.79	45.83	46.06	5.15	0.12	50.00	0.91

Table 3
Adjusted Rand indices for analytic clustering and *k*-means clustering and for the random swap variants of these.

Dataset	Analytic	*k*-Means	Analytic random swap	Random swap
*s*1	0.86	0.87	0.95	0.95
*s*2	0.89	0.89	0.95	0.95
*s*3	0.85	0.87	0.95	0.95
*s*4	0.84	0.86	0.93	0.92

tering vs. 0.1 s for *k*-means). The proposed method can theoretically be applied to large datasets as well, or datasets with a large number of dimensions or clusters. The time complexity is linear with respect to all of these factors. However, in our implementation, we use line search to optimize and use min-function to calculate the nearest centroids, and we have experienced that time consuming increases heavily when these factors increase, and larger datasets are too heavy for this. See the running time comparisons in Table 2. The software used to compute the values in Table 2 is available at http://cs.uef.fi/sipu/soft.

Experiments with the *s*-sets show that the proposed approach leads to similar membership results for the individual data points. Out of the 15 centroids, typically 12–13 are approximately at the same locations and the other two or three at different locations.

4. Conclusions

We proposed a way to form an analytic squared error function. From this function, the partial derivatives can be calculated, and then a gradient descent method can be used to find a local minimum of the squared error. Analytic clustering and *k*-means clustering provide approximately the same result, whereas analytic random swap clustering sometimes gives a better result than random swapping. In *k*-means, there are two phases in one iteration, but in analytic clustering these two phases are combined into a single phase. As a future work, we could consider an implementation including also the gradient calculation and the use of the gradient descent method. Also, then, it would be natural to set a suitable value for the power p, for which now only an extreme theoretical upper limit can be calculated.

References

[1] A. Ahmadi, F. Karray, M.S. Kamel, Model order selection for multiple cooperative swarms clustering using stability analysis, Inform. Sci. 182 (2012) 169–183.
[2] D. Aloise, A. Deshpande, P. Hansen, P. Popat, NP-hardness of Euclidean sum-of-squares clustering, Mach. Learn. 75 (2009) 245–248.
[3] D. Arthur, S. Vassilvitskii, How slow is the *k*-means method?, in: Proceedings of the 2006 Symposium on Computational Geometry (SoCG), pp. 144–153.
[4] D. Arthur, S. Vassilvitskii, k-Means++: the advantages of careful seeding, in: SODA '07: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007, pp. 1027–1035.
[5] L. Bobrowski, J.C. Bezdek, *c*-Means clustering with the l_1 and l_∞ norms, IEEE Transactions on Systems, Man and Cybernetics 21 (1991) 545–554.
[6] J.D. Carroll, A. Chaturvedi, *k*-Midranges clustering, in: A. Rizzi, M. Vichi, H.H. Bock (Eds.), Advances in Data Science and Classification, Springer, Berlin, 1998.
[7] S.G. Chen, P.Y. Hsieh, Fast computation of the *N*th root, Computers & Mathematics with Applications 17 (1989) 1423–1427.
[8] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximun likelihood from incomplete data via the EM algorithm, Journal of Royal Statistical Society B 39 (1977) 1–38.
[9] C.H.Q. Ding, X. He, H. Zha, M. Gu, H.D. Simon, A min–max cut algorithm for graph partitioning and data clustering, in: Proceedings IEEE International Conference on Data Mining 2001 (ICDM), pp. 107–114.
[10] W.H. Equitz, A new vector quantization clustering algorithm, IEEE Trans. Acoust., Speech, Signal Proces. 37 (1989) 1568–1575.
[11] P. Fränti, J. Kivijärvi, Randomized local search algorithm for the clustering problem, Pattern Anal. Appl. 3 (2000) 358–369.
[12] P. Fränti, O. Virmajoki, Iterative shrinking method for clustering problems, Pattern Recogn. 39 (2006) 761–765.

[13] P. Fränti, O. Virmajoki, V. Hautamäki, Fast agglomerative clustering using a *k*-nearest neighbor graph, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1875–1881.
[14] M. Inaba, N. Katoh, H. Imai, Applications of weighted voronoi diagrams and randomization to variance-based *k*-clustering, in: Proceedings of the 10th Annual ACM symposium on computational geometry (SCG 1994), 1994, pp. 332–339.
[15] A. Likas, N. Vlassis, J. Verbeek, The global *k*-means clustering algorithm, Pattern Recogn. 36 (2003) 451–461.
[16] D. MacKay, An example inference task: clustering, in: Information Theory, Inference and Learning Algorithms, Cambridge University Press, 2003, pp. 284–292 (Chapter 20).
[17] J. MacQueen, Some methods of classification and analysis of multivariate observations, in: Proc. 5th Berkeley Symp. Mathemat. Statist. Probability, vol. 1, 1967, pp. 281–296.
[18] D. Pelleg, A. Moore, *X*-Means: Extending *k*-means with efficient estimation of the number of clusters, in: Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 2000, pp. 727–734.
[19] J.M. Peña, J.A. Lozano, P. Larrañaga, An empirical comparison of four initialization methods for the *k*-means algorithm, Pattern Recogn. Lett. 20 (1999) 1027–1040.
[20] H. Späth, Cluster Dissection and Analysis: Theory FORTRAN Programs, Examples, Wiley, New York, 1985.
[21] D. Steinley, *k*-Means clustering: a half-century synthesis, Brit. J. Math. Stat. Psychol. 59 (2006) 1–34.
[22] D. Steinley, M.J. Brusco, Initializing *k*-means batch clustering: a critical evaluation of several techniques, J. Classif. 24 (2007) 99–121.
[23] H.D. Vinod, Integer programming and the theory of grouping, J. Roy. Stat. Assoc. 64 (1969) 506–519.
[24] L. Zhang, Q. Cao, A novel ant-based clustering algorithm using the kernel method, Inform. Sci. 181 (2011) 4658–4672.
[25] Q. Zhao, V. Hautamäki, I. Kärkkäinen, P. Fränti, Random swap EM algorithm for finite mixture models in image segmentation, in: 16th IEEE International Conference on Image Processing (ICIP), pp. 2397–2400.
[26] C. Zhong, D. Miao, P. Fränti, Minimum spanning tree based split-and-merge: a hierarchical clustering method, Inform. Sci. 181 (2011) 3397–3410.