Probability k-means Clustering for Neural Network Architecture

Ka-Hou Chan
School of Applied Sciences
Macao Polytechnic Institute
Macao, China
chankahou@ipm.edu.mo

Wei Ke
Macao Polytechnic Institute
Macao, China
wke@ipm.edu.mo

Sio-Kei Im
Macao Polytechnic Institute
Macao, China
marcusim@ipm.edu.mo

ABSTRACT

Cluster analysis indices are aimed at classifying the elements used to estimate the quality of the categories based on their similarity. It is a challenging task because, with the same data set, there may be many partitions that fit natural groupings of a given data set. Applications of clustering can include pattern recognition, image analysis and information retrieval. We propose an approach based on the k-means concept that clustering centers more often have a higher density than their neighbors: then we use a probability k-means algorithm to achieve fuzzy clustering in continuous form over a relatively large distance from other points with higher densities. Further, in order to follow the mainstream neural network architecture, we define a favorable activation function and corresponding loss function for the clustering iteration. Our method come from the basis of a clustering procedure in which the number of clusters arises intuitively and clusters are achieved regardless of the high dimensions. We demonstrate the result of complete algorithm on several clustering test cases.

CCS CONCEPTS

• **Computing methodologies → Cluster analysis**.

KEYWORDS

Clustering Analysis, probability k-means, Neural Network Architecture

1 INTRODUCTION

Clustering is considered to be one of the cornerstones of unsupervised machine learning. One of the biggest challenges in machine learning and data governance is how to handle the proliferation of data sources and the exponential growth of data, especially in light of the increasing uses of computers and sensor networks. Data mining is now widely used to discover useful or meaningful information in big data. Generally, in order to determine how suitable clustering would be for data mining a given data set, a clustering model is required, and hence determining whether clustering is feasible has the same order of complexity as finding the optimal clustering model. However, clustering is relatively novel in fields such as image segmentation, document analysis, customer segmentation, pattern recognition and information retrieval [1, 2] and to get optimal model parameters is relatively time-consuming.

Meanwhile, with the improvement in computational power and increasing popularity of neural networks, many studies have taken place on the architecture of neural networks [3, 4]. One of the major advancements in deep learning is using the activation function and the corresponding loss function. The activation function will always project the input data into an unknown hidden (domain) layer but the projected result can be comparable and combinable, then the loss function will calculate the current error/cost value about the whole network states. This error is considered as an evaluation which will feedback to the learning parameters to make adjustment and prepare for the next evaluation (Such as activation function *softmax* and loss function *cross-entropy* are the well-known combination in convolutional neural network [5]). As the problem size increases, the computational cost grows larger, so hindering further development and proper scaling of the neural network. To overcome these challenge, many algorithms have been introduced for optimizing the neural network structure and weights-liked evolutionary algorithms [6, 7]. Hence, the definition of the clustering criterion can be arbitrary and should be informed by the specific application for specific case.

In this paper, we implement the k-means approach into the neural network. The clustering processes must be divided into several parts, which must satisfy the mainstream architecture. A major challenge is that, because the best approximation of the neural network is obtained through continuous iterations, we have to achieve continuity of clustering; to do this we propose a novel probabilistic approach to make this problem continuous. Once given a competitive learning based clustering method, we then must define the learning parameter which can find the clustering center and be adjusted by feedback circuit, and the system will update the probability after the learning phase is completed.

2 RELATED WORKS

Cluster analysis is a fundamental machine learning method that aims to partition data set into homogeneous groups [1, 8]. Different from classification methods in supervised machine learning [5], clustering does not require external labels to identify the sample data. One of the most well-known clustering algorithms is the k-means that aims to find the clustering centers by averaging the distance of the assigned data, which has a loose relationship to the k-nearest neighbor classifier [9]. Depending on the inherent structure and characteristics, similar data will be partitioned into the same clusters, which ensures that data in a cluster is more similar to other data in the same cluster than that in other clusters. Assume a data set $X = \{x_1, \cdots, x_N\}$, where $x \in R$ can be a d-dimensional

ICAAI 2019, October 26–28, 2019, Istanbul, Turkey

ACM ISBN 978-1-4503-7253-4/19/10…$15.00
https://doi.org/10.1145/3369114.3369147

real vector, the M-clustering problem aims at partitioning this data set into M clusters $C_1, \cdots, C_M$. A widely function used clustering is to minimize the within-cluster sum of squares error (also called clustering error) and depends on the mean as follows:

$$E(\mu_1, \cdots, \mu_M) = \sum_{i=1}^{M} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{1}$$

where μ_i is the mean of data in C_i, the clustering error E is the sum of variance of all subsets. The k-means algorithm will find locally optimal solutions with respect to the E. It is a fast, iterative algorithm that has been used in many clustering applications. Further, the concepts of partitioned clustering have been extended to the search for more complex data structures, leading to the notion of fuzzy and possibility [10]. By using an iterative approach and re-computing, the clustering centers can attempt to find the best partitioning. The advantage of k-means is that it is a simple and efficient algorithm, but it is sensitive to outliers and its non-global view makes it highly dependent on the domain of initialization which may cause to reach only a local optimum [11]. However, k-means-type algorithms are widely used in clustering tasks and data mining (which is another way of studying a clustering task). The k-means clustering first assigns each data sample to the nearest cluster using a distance/similarity measurement, and then updates the centroid of each cluster to reassign the data points.

As an improvement, an advanced method of extending the k-means algorithm is known as fuzzy k-means [12], and then modified as the general fuzzy clustering approach by [13, 14]. A robust fuzzy algorithm was proposed in [15] that uses a kernel function instead of the multidimensional distance, and also employs center initialization to save computational cost. [16] is closely related to the k-means clustering, which can simultaneously minimize the within cluster dispersion and maximize the negative weight entropy to determine clusters objects. Many other modifications have been proposed to improve the performance by applying new clustering techniques. [17] supplemented the fuzzy-k-means loss function with a penalty term, resulting in spatially-smooth membership functions and [18] presented the incorporation of a neighborhood additive term into the loss function, thus creating the bias-corrected fuzzy-k-means algorithm. Later [19] introduced an adaptive method to obtain the weight of local spatial information in the loss function, thereby reducing noise and dealing with classification. Further, a self-organizing clustering network was proposed in [20] that is not based on the minimization of any known loss function, which raised an interesting concept, but left a few major issues such as non-gradient convergence and non-optimized procedure. The model outputs are Euclidean distance and the probability can be related to the distance using suitable formula, so lowering considerably the computation time required to perform clustering.

3 CLUSTERING NEURAL NETWORK

In order to map the clustering algorithm into the neural network architecture, we have to determine how to make use of learning parameters in the k-means algorithm because k-means-type clustering algorithms are designed to find clusters that fit some static models and rely on the correct choice of parameters for capturing cluster characteristics. We must find the clustering center as a reference point to result a clustering error at every iteration. On the other hand, the clustering error E can also be considered as a loss function in the neural network, but Eq. 1 indicates that the membership between data and clusters are discrete and finite, which is hard to describe the transition process between any different clusters if partitions are discrete.

3.1 Learning Parameters Definition

To review the k-means algorithms, we define M vectors as learning parameters for the entire training and prepare $S = \{s_1, \ldots, s_M\}$ as the dataset, here s_i is a d-dimensional real vector thus there is a total of $M \times d$ learning parameters. Following the clustering error concept like Eq. 1, s_i can be seen as the clustering center for the cluster i. We determine which cluster the data belongs to based on their distance as

$$\arg\min_i \|x - s_i\| \tag{2}$$

Different to the mean value μ_i, s_i is dynamic and is only adjusted by the iterative optimizers (like SGD, Adam) and learning rate in the neural network. These learning parameters will process the input data then return the clustering error, which acts as an evaluation value that will be fed back to the learning parameters to make adjustment and prepare for the next evaluation. After enough iterations, the ultimate goal of this should be:

$$\lim_{t \to \infty} s_i^{(t)} \to \mu_i$$

Here (t) is the number of iterations and $i \in \{1, \cdots, M\}$. There is an issue about the concept: s_i is a single vector but μ_i is a composite vector. It often causes a local optimum to be reached if our algorithm bypasses consideration of mean μ_i. Thus the distance between data and learning parameters cannot be applied in Eq. 1 directly. However, we can make further use of these distances between every data point and each cluster in the next operation.

3.2 Probability Clustering

As discussed in traditional k-means, each data point can belong to only one set. We would like to make use of a probability function that can make this membership function fuzzy as $P_C(x)$, which represents the probability of data x belonging to cluster C. This probability is assigned a positive real number between 0.0 and 1.0. Since the element is certain to belong to one or more clusters, the total probability is 1.0 as follow:

$$\sum_{i=1}^{M} P_{C_i}(x) = 1.0 \tag{3}$$

This function can take care of all membership relations between each data point and every cluster, and changes can be achieved by iterative simulation for the approximated continuously. Thus, each data point has a probability of belonging to clusters rather than belonging to just one cluster, and the clustering center of a cluster is the mean of all points, weighted by their probability of belonging to the cluster.

3.3 Membership as Activation Function

Continuing the previous two subsections, we now have got the distances between every data value and each clustering center as $d_i = \|x - s_i\|$, and the distance is related to the inverse probability of belonging:

$$P_{S_i}(x) = \frac{1}{\|x - s_i\|} = \frac{1}{d_i} \tag{4}$$

It can be found that data x closest to the clustering center s_i is given a greater value than the others. Corresponding to Eq. 2, the membership can be considered as:

$$\arg\max_i P_{S_i}(x) \tag{5}$$

According to the condition in Eq. 3, Eq. 4 must be normalized before minimizing the clustering error. We consider that there are several potential approaches that can be used for this purpose:

Well-known activation function *softmax* the larger input components will correspond to slightly larger probabilities, and other input components will result in slightly smaller probabilities.

$$P_{C_i}(x) = \frac{e^{P_{S_i}(x)}}{\sum_{j=1}^{M} e^{P_{S_j}(x)}}$$

Percentage of Squared Euclidean Norms (*PSEN*) it is also a non-linear function but the larger input components will correspond to larger probabilities, and other input components will result in very small probabilities compared with softmax.

$$P_{C_i}(x) = \frac{P_{S_i}(x)^2}{\sum_{j=1}^{M} P_{S_j}(x)^2}$$

These functions take as input a set of real numbers, and normalize that into a probability distribution in which each component will be in the range $(0.0, 1.0)$. As an example, take an input of distance as

$$\mathbf{d} = [40.66, 90.49, 4.42, 63.69, 39.31, 105.90]$$

the probabilities P_{C_i} for *softmax* and *PSEN* will become

$$softmax\left(\frac{1}{\mathbf{d}}\right) = [0.162, 0.159, 0.198, 0.160, 0.162, 0.159]$$

$$PSEN\left(\frac{1}{\mathbf{d}}\right) = [0.011, 0.002, 0.968, 0.005, 0.012, 0.002]$$

The components sum up to 1.0, so that they can be interpreted as probabilities (see Fig. 1). Consider their gradient as follows,

$$\nabla softmax(\mathbf{P_C}) = \left[softmax\left(P_{C_i}\right)\left(1 - softmax\left(P_{C_i}\right)\right)\right]_{1 \le i \le M} \tag{6}$$

$$\nabla PSEN(\mathbf{P_C}) = \left[\frac{2}{P_{C_i}} PSEN\left(P_{C_i}\right)\left(1 - PSEN\left(P_{C_i}\right)\right)\right]_{1 \le i \le M} \tag{7}$$

The probabilities results are shown in Fig. 1 and their corresponding curves in Fig. 2.

The *softmax* usually causes some very big or very small clusters. Because its gradient becomes very small when the data is far away from the clustering center, there is almost no further change to the prediction, which is a vanishing gradient case. The result is the network will to learn further, or will be too slow to reach an accurate prediction. The *PSEN* gradient depends on the inverse

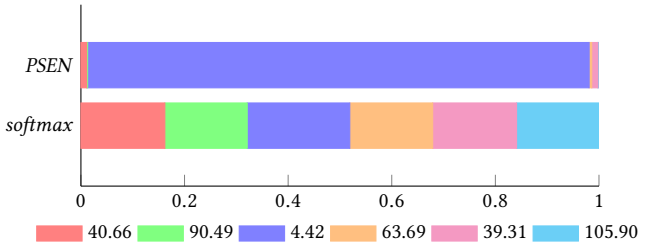


Figure 1: The probabilities are resulted by the *softmax* and *PSEN* respectively.

ratio $\frac{2}{P_{C_i}}$; this can smooth the vanishing gradient problem denoted by one high and other very small probability.

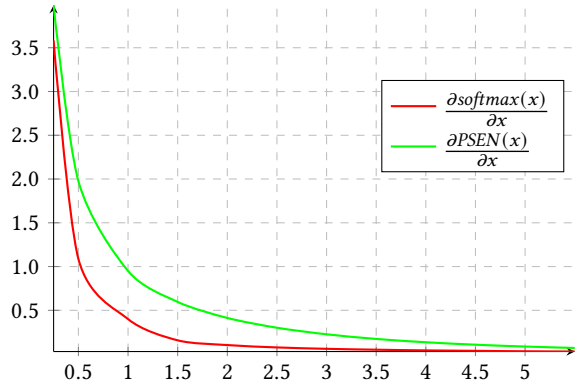


Figure 2: The trend of gradient about *softmax* and *PSEN*.

As shown in Fig. 2, it is found that the vanishing gradient problem can be effectively alleviated by using the *PSEN*, so allowing the learning parameters that are far away from the clustering center to contribute. Therefore, we use the *PSEN* as the ending output layer in our architecture.

3.4 Clustering Error as Loss Function

To evaluate current learning parameters, we make use of the weighted function with clustering analysis to minimize the clustering error, and then the mean μ_i in Eq. 1 will become:

$$\mu_i = \frac{\sum_{x \in X}\left(P_{C_i}(x) \cdot x\right)}{\sum_{x \in X} P_{C_i}(x)} \tag{8}$$

$$E(\mu_1, \cdots, \mu_M) = \sum_{i=1}^{M} \frac{\sum_{x \in X}\left(P_{C_i}(x) \cdot \|x - \mu_i\|^2\right)}{\sum_{x \in X} P_{C_i}(x)} \tag{9}$$

The learning parameters are the estimator that minimizes the expected loss experienced under the weighted squared-error loss function. This clustering error can provide the result of loss function while learning parameters run on the clustering structure. The number of iterations until convergence is often stable, and results only improve slightly after several iterations.

4 IMPLEMENTATION

Our network uses feedforward architecture [21, 22]. A simple kind of neural network is a single-layer perceptron network, which consists of a single layer of output nodes (see Fig. 3). In order to achieve our algorithm, we make use of the scientific deep learning framework called PyTorch [23] for our implementation and all our experiments were conducted on an Nvidia GeForce GTX 1080 with 8.0GB of video memory.

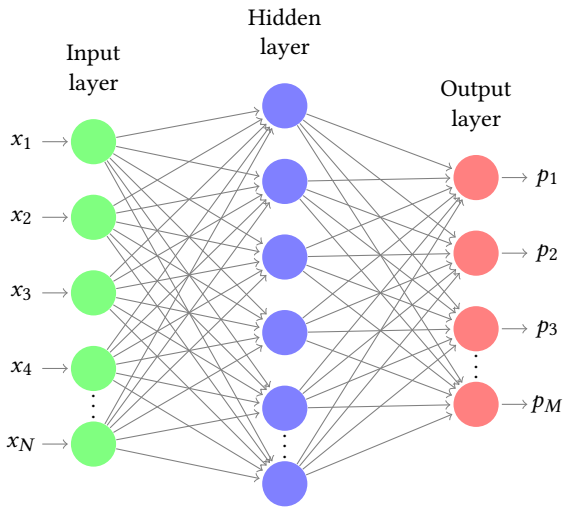


Figure 3: A simple feedforward architecture, input layer followed by fully-connected hidden layer, ending with a *PSEN* output.

4.1 Overall Architecture

As discussed in Section 3.1, there are $M \times d$ learning parameters as clustering center we must allocate for the training algorithm, and there are $N \times d$ non-clustered data for input. Then using the approach of Section 3.2 and 3.3 we can find the distance and its probabilities. This processing can be seen as a hidden-layer output with an activation function. This computation can be finished in the *forward* function (that is an implementation interface provided by PyTorch). We define the network structure as in Algorithm 1.

Algorithm 1 Module construct of Neural Network

function INIT
  $S \leftarrow$ learning parameters initialization
end function

function FORWARD($S \leftarrow \{s_1, \cdots, s_M\}, X \leftarrow \{x_1, \cdots, x_N\}$)
  $d_{i,j} = \left\| s_i - x_j \right\|$
  $p_i = PSEN\left(d_i\right)$ ▷ Activation Function
  return $P \leftarrow \{p_1, \cdots, p_M\}$
end function

Section 3.4 presented the loss function that calculates the clustering error in Algorithm 2 and uses its result to make an adjustment of the learning parameters. They also accept the gradients of these outputs and return the gradients of the inputs (formally, the product including the term for their respective operation). The products passed around are themselves variables, making the evaluation of the graph differentiable.

Algorithm 2 Clustering Error as Loss Function

function FORWARD($P \leftarrow \{p_1, \cdots, p_M\}, X \leftarrow \{x_1, \cdots, x_N\}$)
  $\mu_i = \dfrac{\sum_{j=1}^{N} \left(p_{i,j} \cdot x_j\right)}{\sum_{j=1}^{N} p_{i,j}}$
  $E\left(\mu_i\right) = \dfrac{\sum_{j=1}^{N} \left(p_{i,j} \cdot \left\| x_j - \mu_i \right\|^2\right)}{\sum_{j=1}^{N} p_{i,j}}$
  $loss = \sum_{i=1}^{M} E\left(\mu_i\right)$
  return *loss*
end function

The goal is to minimize the clustering errors thus implementing a form of iterative optimizers by Adam. This program instructs PyTorch to perform an immediate operation on *loss*. Further, it could be used in the backward computation (recall that the first derivative function is in Eq. 7).

4.2 Result and Discussion

We present the results of comparative experiments with different datasets available form clustering datasets:

Varying the number of clusters [24]: The data sets A_1 to A_3 are 2-dimensional sets with a varying number of circular clusters (M equal to 20, 35, 50) as illustrated in Fig. 4.

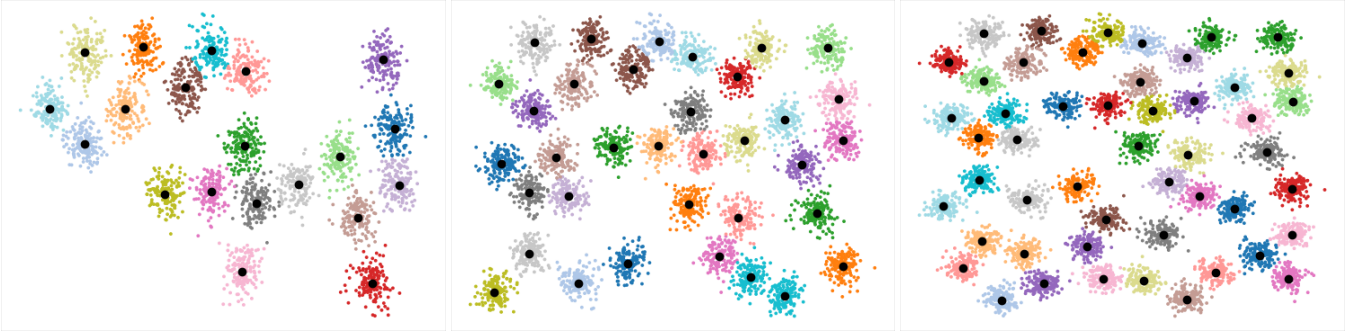
Varying spatial complexity [25]: The data sets S_1 to S_4 are 2-dimensional sets with varying complexity in terms of spatial data distributions, as shown in Fig. 5 with 15 clusters.

These input data are recommended to be scaled within $(0.0, 1.0)$ before training, the initial learning rate is set to 0.05 and the result is from the original view test without any enhancement. Note that the initialization of learning parameters about the clustering center should surround the mean of the dataset. For more details, the complete python source code can be found at:

`github.com/ChanKaHou/Clustering-Neural-Network`

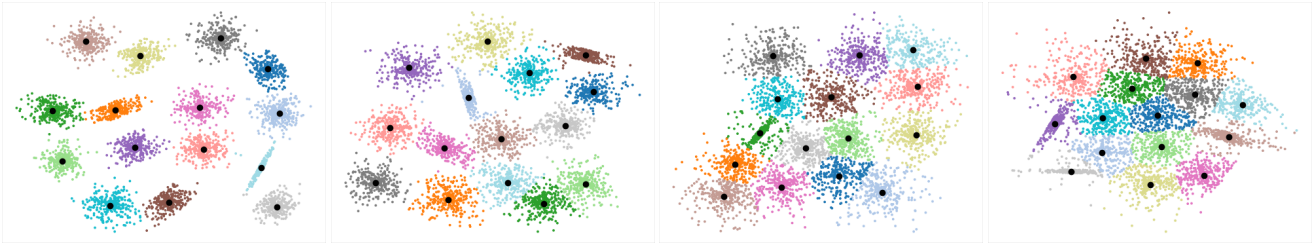
We first analyze the convergence speed of each individual data set. Our experiments processed 130 iteration steps for each data set. We conducted our experiments on both clustering performance and a median of 10 turns. As shown in Fig. 6 and 7, the convergence speed is very fast and then becomes stable after 60 iteration steps. This is mainly due to our initialization method, otherwise, some clustering centers would always be excluded by being far away. In the meantime, the clustering error increases with the increase of M, often generating two (and no more than two) cluster centers within the ground truth partition. This problem is hard to avoid because it is used as a local optimal method, but will not be overlapped (such as S_3 and S_4).

In all fairness, since our experiment models do not use any optimization method, *e.g.*, improved initialization algorithm, the results



(a) data set A_1 (b) data set A_2 (c) data set A_3

Figure 4: 2-dimensional data sets with varying number of circular cluster. These data sets have 3000, 5250 and 7500 vectors scattered across 20, 35 and 50 predefined clusters.



(a) data set S_1 (b) data set S_2 (c) data set S_3 (d) data set S_4

Figure 5: 2-dimensional data sets with different complexity in spatial data distribution. These data sets have 5000 vectors scattered across 15 predefined clusters with varying degrees of overlap.

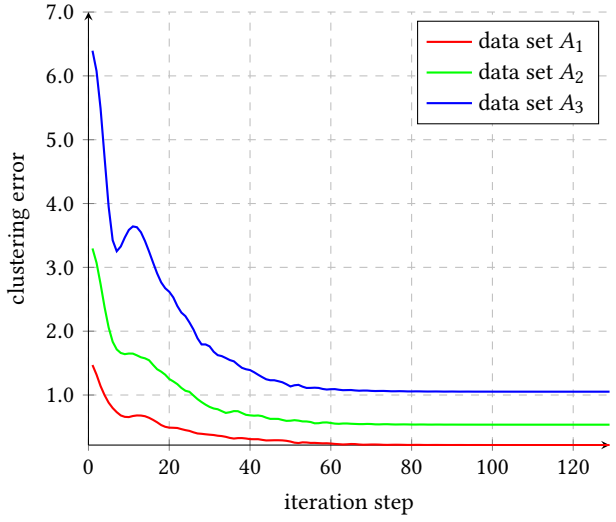


Figure 6: The trend of clustering error with 130 iteration steps for data set A.

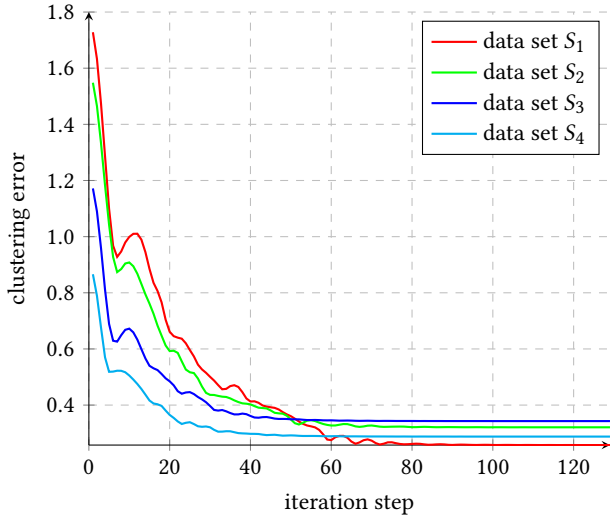


Figure 7: The trend of clustering error with 130 iteration steps for data set S.

of the traditional k-means implementation should be worse than other works, but our proposed probability k-means consistently outperform others on clustering without the optimization processing.

5 CONCLUSION

In this paper, we introduce a probability k-means clustering algorithm that is implemented with a neural network architecture for finding the clustering center located objects. We clarify the definition of the local optimality of the clustering method and show that if the correct number of algorithms is given, the algorithm can find all the test data clusters. The initialization does not lead the algorithm into a local optimum, and because of the feedforward architecture, the convergence speed is significantly faster while the initialization of clustering center should surround the mean of the dataset. To evaluate our algorithm, five real vector data sets were used in experiments. The results show that overlapping cases can be correctly partitioned, but other cases can be trapped in a local optimum. Our algorithm found that clusters in larger datasets are faster and have the same accuracy as traditional implementations, and this cluster analysis is a suitable algorithm for large datasets.

ACKNOWLEDGMENT

This work was Funded by The Science and Technology Development Fund, Macau SAR (File no. 0001/2018/AFJ)

REFERENCES

[1] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[2] John D Kelleher, Brian Mac Namee, and Aoife D'arcy. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. MIT Press, 2015.

[3] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

[4] Richard P Lippmann. An introduction to computing with neural nets. *IEEE Assp magazine*, 4(2):4–22, 1987.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Eysa Salajegheh and Saeed Gholizadeh. Optimum design of structures by an improved genetic algorithm using neural networks. *Advances in Engineering Software*, 36(11-12):757–767, 2005.

[7] Guochu Chen and Jinshou Yu. Particle swarm optimization neural network and its application in soft-sensing modeling. In *International Conference on Natural Computation*, pages 610–617. Springer, 2005.

[8] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*, 2(3):283–304, 1998.

[9] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[10] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[11] Fengqin Yang, Tieli Sun, and Changhai Zhang. An efficient hybrid data clustering method based on k-harmonic means and particle swarm optimization. *Expert Systems with Applications*, 36(6):9847–9852, 2009.

[12] James C Bezdek. A convergence theorem for the fuzzy isodata clustering algorithms. *IEEE transactions on pattern analysis and machine intelligence*, (1):1–8, 1980.

[13] M-S Yang. A survey of fuzzy clustering. *Mathematical and Computer modelling*, 18(11):1–16, 1993.

[14] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[15] SR Kannan, S Ramathilagam, R Devi, and Evor Hines. Strong fuzzy c-means in medical image data analysis. *Journal of Systems and Software*, 85(11):2425–2438, 2012.

[16] Jinglin Xu, Junwei Han, Kai Xiong, and Feiping Nie. Robust and sparse fuzzy k-means clustering. In *IJCAI*, pages 2224–2230, 2016.

[17] Dzung L Pham. Fuzzy clustering with spatial constraints. In *Proceedings. International Conference on Image Processing*, volume 2, pages II–II. IEEE, 2002.

[18] Mohamed N Ahmed, Sameh M Yamany, Nevin Mohamed, Aly A Farag, and Thomas Moriarty. A modified fuzzy c-means algorithm for bias field estimation and segmentation of mri data. *IEEE transactions on medical imaging*, 21(3):193–199, 2002.

[19] Ze-Xuan Ji, Quan-Sen Sun, and De-Shen Xia. A modified possibilistic fuzzy c-means clustering algorithm for bias field estimation and segmentation of brain mr image. *Computerized Medical Imaging and Graphics*, 35(5):383–397, 2011.

[20] Teuvo Kohonen. *Self-organization and associative memory*, volume 8. Springer Science & Business Media, 2012.

[21] Andreas Zell. *Simulation neuronaler netze*, volume 1. Addison-Wesley Bonn, 1994.

[22] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[24] Ismo Kärkkäinen and Pasi Fränti. *Dynamic local search algorithm for the clustering problem*. University of Joensuu, 2002.

[25] Pasi Fränti and Olli Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–775, 2006.