SOM++: Integration of Self-Organizing Map and K-Means++ Algorithms

Yunus Dogan, Derya Birant, and Alp Kut

Dokuz Eylul University, Department of Computer Engineering,
Tinaztepe Campus, Buca, 35397 Izmir, Turkey
{yunus,derya,alp}@cs.deu.edu.tr

Abstract. Data clustering is an important and widely used task of data mining that groups similar items together into subsets. This paper introduces a new clustering algorithm SOM++, which first uses K-Means++ method to determine the initial weight values and the starting points, and then uses Self-Organizing Map (SOM) to find the final clustering solution. The purpose of this algorithm is to provide a useful technique to improve the solution of the data clustering and data mining in terms of runtime, the rate of unstable data points and internal error. This paper also presents the comparison of our algorithm with simple SOM and K-Means + SOM by using a real world data. The results show that SOM++ has a good performance in stability and significantly outperforms three other methods training time.

Keywords: Data Mining, Clustering, Self-Organizing Map, K-Means++, Mining Methods and Algorithms.

1 Introduction

Cluster analysis is the process of grouping data into subsets such that each item in a cluster is more similar to the items in the same cluster than to the other items at the outside of the cluster. Generally, distance measures like Euclidean distance, Manhattan distance are utilized to evaluate the dissimilarity between data points. Cluster analysis is one of the most useful tasks in machine learning and data mining, and has been used in a variety of fields such as marketing, banking, medicine and telecommunication. It has been widely used in dimensionality reduction, information extraction, density approximation and data compression [15] [6] [7] [16].

The K-means [12] algorithm is the most commonly used partitioning cluster algorithm with its easy implementation and its efficient execution time. Self-organizing map (SOM) [11] is an unsupervised, well-established and widely used clustering technique.

In SOM, initial weight values are assigned randomly, method performance is sensitive to these values and it is prohibitively slow in large data applications. In order to decrease the time complexity of SOM, we investigated different initialization procedures for optimal SOM and now propose K-Means++ as the most convenient method, given the proper training parameters.

P. Perner (Ed.): MLDM 2013, LNAI 7988, pp. 246–259, 2013.

K-Means++ algorithm gives more successful results than standard K-Means at accuracy and consistency [3]. Because, the K-Means algorithm works only to find a local optimum and this local optimum often becomes poor by using random initial center points; however, K-Means++ starts with rational initial points, thus K-Means++ approximates the best clustering space. Also, K-Means++ outperforms in speed, too. K-Means++ guarantees O(logk) as the complexity time; however, K-Means has a complexity time as $O(n^{kd+1} \log n)$, where k is the number of clusters, n is the number of data and d is the Euclidean distance between two clusters [3].

This paper proposes a new clustering algorithm SOM++, which is composed by K-Means++ method followed by SOM clustering. The algorithm consists of two stages: First, using K-Means++ method to determine the initial weight values instead of assigning in randomly, then clustering task is done in the second stage by SOM as an unsupervised clustering method. The experimental results show that the proposed algorithm, SOM++, is considerably better than the conventional SOM based algorithms in terms of runtime, the rate of unstable data points and internal error. It generates similar clustering results with other SOM based clustering algorithms, however the use of it requires the smaller training time.

The rest of this paper is organized as follows. The second section reviews the literature and describes SOM, K-Means++ algorithm and how SOM and K-Means were combined in the previous studies. After the new clustering algorithm SOM++ is explained in detail and also the principle and the architecture of the algorithm are presented in the third section, we demonstrate how the proposed algorithm can be applied on a real world data, including dataset description, experimental setup, and performance analysis and clustering results in the fourth section. The experimental studies indicate with figures of map combinations and tables of error rates. Section 5 points out the comparison of the results of SOM++, simple SOM, SOM+K-Means and SOM+K-Means++ (include all phases of K-Means++). Finally, the summary, conclusions and future work are given in the Section 6.

2 K-Means++ & SOM

The complexity of SOM algorithm is O(NC), where N is the input vector size and C is the number of dataset presentation cycles. N contains n^2w as the multiply of the map size n^2 and the number of weights w. C contains n^2a as the multiply of the map size n^2 and the number of attributes a. The number of attributes is equal to the number of weights; therefore, the complexity of SOM algorithm obtains $O(N^2)$[14].

While simple SOM has been previously used in many applications, extended versions of SOM has also been proposed such as FSOM (Fast SOM) [15], ABSOM (Ant Based SOM) [6], and ESOM (Emergent SOM) [7].

When the performance of the K-Means++ algorithm were evaluated on four datasets and K-Means++ consistently outperformed K-Means, both by completing faster and by achieving a lower potential value. For example, on one dataset, K-Means++ terminates almost twice as fast while achieving potential function values about 20% better, on the larger dataset, it is obtained up to 70% faster and the potential value is better by factors of 10 to 1000. For this reason, we propose K-Means++ algorithm in this paper, instead of K-Means [3].

In recent years, several studies have compared different SOM-based two-stage methods. For instance, while Souza et al. [16] compared SOMK (SOM+K-Means) with SOMAK (SOM+Ant K-Means), Chi and Yang [5] compared both ABSOM (Ant-based Self-Organizing Map) with Kohonen's SOM individually, and SOM+K-Means with ABSOM+K-Means. Besides, Chiu et al. [7], [8] compares four approaches simple K-Means, SOM+K-Means, PSO+K-Means (Particle swarm optimization (PSO)) and PSHBMO (Particle Swarm Optimization with Honey Bee Mating Optimization). As another example, the study of Corrêa and Ludermir [9] about the comparison of several SOM-based two-stage approaches: SOM, SOM+KM (K-Means), SOM+W.KM (Weighted K-Means), SOM+AY (proposed by Azcarraga and Yap) and SOM+W.AY (Weighted AY Method), in terms of classification accuracy and runtime.

Recently, performing K-Means method after usage of SOM was also studied for different purposes as examples of the emergency planning to deal with extreme events such as earthquake, flood and fire [2], clustering meteorological data [10], the biological wastewater treatment process [1], and the identification of day types of electricity load [4]. Our proposed algorithm, SOM++ is a general clustering algorithm; as a result, it can be used in many different applications for different purposes.

To the best of our knowledge, however, this paper is the first in performing K-Means method before training neurons by usage of SOM to determine the initial weight values of SOM.

3 Methodology

The study of Su, Liu and Chang shows that the initializing the weight values increases the performance of SOM [17]. SOM++ shows that initializing the weight values by K-Means++ (without K-Means clustering) increases the performance of SOM. Also, the study of Attik, Bougrain and Alexandre mentions that initializing the weight values by K-Means clustering increases the performance of SOM without any example or method [3]; SOM++ is a supportive and integral study of these studies with K-Means++ (without K-Means clustering) and a new sequential assignment algorithms.

In this section, it is explained that our new algorithm SOM++, a two-stage clustering algorithm uses the combination of two data mining techniques, namely SOM and K-means++ clustering. SOM algorithm uses neurons for all points on its map and these neurons have weight values for all attribute values. Before showing the details of SOM++ algorithm, these weight values are indicated in the following part. In SOM, input neurons are fully connected to output neurons, and each connection has a weighting value. In the initialization process of SOM, each neuron is associated with a random weight vector (wi = wi1, wi2, …, win), which has the same dimension (n) as the input vector (xi = xi1, xi2, …, xin). Using the Euclidean measure, distance between the input vector and the incoming weight vector of each output map neuron is calculated. The output neuron with the smallest distance is declared the winner.

After that, neuron weights are subsequently updated according to (1) using a neighborhood function (2), which minimizes the overall distance between the neuron itself and its neighbors.

$$wij(t + 1) = wij + h(t)(xi - wij) \quad (1)$$

where wij(t) is the connection weight from input i to output neuron j at time t; xi is element i of input vector x, and h is the neighborhood function.

$$h(t) = \alpha GF \quad (2)$$

where α is the learning rate; GF is the Gaussian Function (3).

$$GF = exp\ (-\sum \| wui - cui \| 2/2\phi 2\ (t)) \quad (3)$$

where ϕ is the neighborhood width parameter and GF uses the Euclidean distance between the winner unit (wu) and the current unit (cu).

SOM method performance is sensitive to the randomly assigned initial weight values and it is prohibitively slowed in the large-scale applications. In order to decrease the time complexity of SOM, this paper proposes K-Means++ to determine the initial weight values, instead of random process. In this approach, K-Means++ centers are assigned as SOM weight values; thus, SOM will require fewer iterations. Since the K-means algorithm is more computationally efficient than SOM, the general solution will be faster.

3.1 Description of SOM++ Algorithm

The proposed SOM++ is a two-stage algorithm, which is a combination of SOM and the initialization method of centers in K-means++. Fig. 1 and Fig. 2 show pseudo codes for SOM++ algorithm. The algorithm starts by finding the center points for the all clusters by using the method of K-Means++ which initializes the centre points of the clusters. There must be two sets named as Φ and D. Set D collects the centers of all clusters during the first part of SOM++ (step 1 and step 6 in Fig. 1). Set Φ collects the distances between each data and each center (step 2 in Fig. 1). The distances are calculated by using the Euclidean Distance. According to sum of squares of distances in set Φ, the centers are obtained (step 3-4-5 in Fig. 1). After obtaining k centers in set D (step 7 in Fig. 1), the second part of SOM++ is started (Fig. 2).

After these steps, all centre points for all clusters are collected in a set Φ. These centers have the attribute values and these values must initialize the weight values of neurons on the map of SOM++. However, the most suitable method must be decided for locating these initial weight values. If the locating method is not suitable according to the distances of neurons, the result of SOM++ is not different from the result of the standard SOM with the random initialization (Comparing the error rates is given in Section 4.4). Therefore, a new sequence assignment algorithm is implemented by considering the distances between K centers in set Φ.

Fig. 2 and Fig. 3 show the pseudo code of this sequence assignment algorithm.

Firstly, the most different point in set D is calculated (step 1 in Fig. 2) and according to the Euclidean Distance, a sorting operation is done by comparing to this

outlier point. At the end of sorting operation, a new set which has sorted points according to the least similar point is obtained (step 2). The values in these points are assigned to the weight values of neurons as the initial values for SOM++ algorithm like the sequence in Fig. 3 (step 3 in Fig. 2).

	Finding initial weight values of neurons (K-Means++ part)
1	**Select** data from the data set β randomly **Add** this data into a set D
	$D \leftarrow \beta_{Random(0,i)}$
2	**For each** data i in the dataset β **For each** data j in the dataset D **Find** the Euclidean Distances between βi and Dj **Add** the minimum distance into set Φ
	$\forall i \left(\Phi \leftarrow min \left(\forall j \left(\sum_{k=0}^{n} \left\| \beta ik - Djk \right\|^2 \right) \right) \right)$, *where n is equal to the number of attributes, i is equal to the number of data β and j is equal to the number of data in D*
3	**Find** the sum of squares S for the values in Φ
	$S = \sum_{k=0}^{i} \Phi k^2$, *where i is equal to the number of data in Φ*
4	**Select** a real number R between 0 and S randomly
	$R = Random(0, S)$
5	**Find** the unique integer q so that $1^2 + 2^2 + \ldots + q^2 >= R > 1^2 + 2^2 + \ldots + (q-1)^2$
	$\sum_{k=1}^{q} k^2 \geq R > \sum_{k=1}^{q-1} k^2$, *where $q \in \mathbb{N}^+$, $q > 2$*
6	**Add** q^{th} data in β into D
	$D \leftarrow \beta_q$
7	**Repeat** steps 2-3-4-5-6 **until** the number of data in D is equal to the number of clusters K

Fig. 1. The K-Means++ part of SOM++ algorithm

Before training operations in SOM++, the number of iteration is initialized as the number of total data (step 4 in Fig. 2), because actually, the neurons on the map of SOM++ become trained at the beginning; therefore, the number of iteration must not start zero. The advantages of initializing both the number of iteration and weight values of neurons with the values coming from K-Means++ are shown in the Section 4 and 5 in detail. Finally, training operations of neurons starts by using the standard SOM algorithm (step 5 in Fig. 2).

The weight values become close to the final and decisive values by means of K-Means++; on the other hand, it is not enough singly, because the single aim of SOM algorithm is not to do clustering of data correctly. It is also a mapping algorithm; therefore, the places of the clusters win the importance in SOM algorithm. If locating of the weight values which come from the initializing method of K-Means++ is done randomly, the correct neuron cannot have the correct weight values, and the success of SOM++ algorithm is not realized certainly.

In Section 4, the importance of the sequential assignment is tested in detail.

Our sequential assignment algorithm needs the sorted values according to the least similar value to each other in the set. Then, locating operation starts from the neuron in [0, 0] which is the one of the furthest four neurons ([0, 0], [0, (n-1)], [(n-1), 0] and [(n-1), (n-1)] as n^2 is the number of neurons) from the center on the map, because the top element in sorted values is the least similar value. The next values after the least similar value are located like the sequence in Fig. 3.

Initializing weight values of neurons and the number of iteration	
1	**Find** the least similar center L to the other centers in D
	$$L = max\left(\forall i \left(\forall j \left(\sum_{k=0}^{n} \left\| Dik - Djk \right\|^2 \right) \right) \right) ,$$ *where n is equal to the number of attributes, i is equal to the number of data in D and j is equal to i+1 for each i.*
2	**Sort** the centers in D from the most similar center to the least similar center according to L by using the Euclidean Distance **Collect** these centers in θ
	$$\forall i \left(\begin{array}{c} \theta \leftarrow L, \\ L \notin D \\ L = Dmin\left(\forall i \left(\sum_{k=0}^{n} \left\| Lk - Dik \right\|^2 \right) \right), \\ \theta \leftarrow L \end{array} \right) ,$$ *where n is the number of attribute and i is equal to the number of data in D*
3	**Initialize** attribute values of these sorted centers into the weight values of neurons on the map sequentially like the sequence in Fig 3.
4	**Initialize** the number of iteration as the number of the data
	$$I = N ,$$ *where N the number of* data in the dataset *β and I is the iteration number of the standard SOM algorithm*
5	**Start** the standard SOM algorithm

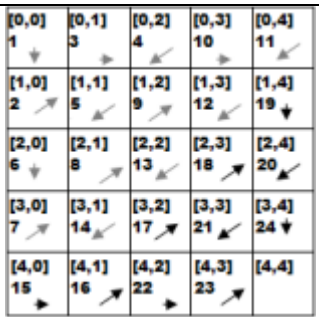
Fig. 2. Initializing parameters part of SOM++ algorithm

The steps on the left and right sides are done to down and inner-cross directions. The steps on the top and bottom sides are done to left and inner-cross directions; however, the steps on the center side are done mix-cross directions. Finally, the furthest values are located in the furthest neurons on the map.

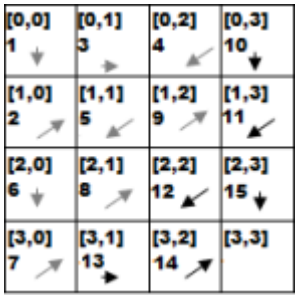
4 Experimental Studies

The error rates and stability of the maps are tested by a dataset with 699 vectorial tuples. These datasets have an attribute which puts the correct classes of all data. However, the datasets are used without this attribute while SOM clustering. Because SOM is a clustering algorithm and does not need a target attribute while training neurons. This attribute is used while testing the stability of the maps.

1	$x = 0, y = 0, n = 0, \forall i\ (M[x,y].W[i] = \theta[n].W[i])$, *where i is the number of the weights in a neuron, M is the map matrix, W is the weight array in a neuron, x and y are the indexes of M, and n is the index of the* θ
2 ↓	$y < BY \Longrightarrow (y = y + 1,\ \ n = n + 1,\ \ \forall i\ (M[x,y].W[i] = \theta[n].W[i]))$, *where BY is the boundary of y in M*
3 ↗	$\exists x,y[\ (y < BY\ \Lambda\ x < BX) \Longrightarrow$ $(x = x + 1,\ \ y = y - 1,\ \ n = n + 1,\ \forall i\ (M[x,y].W[i] = \theta[n].W[i])]$, *where BX is the boundary of x in M*
4 →	$x < BX \Longrightarrow (x = x + 1,\ \ n = n + 1,\ \ \forall i\ (M[x,y].W[i] = \theta[n].W[i]))$, *where BX is the boundary of x in M*
5 ↙	$\exists x,y[\ (x < BX\ \Lambda\ y < BY) \Longrightarrow$ $(x = x - 1,\ \ y = y + 1,\ \ n = n + 1,\ \forall i\ (M[x,y].W[i] = \theta[n].W[i])]$
6	**Repeat** 2-3-4-5 **until** $x = \begin{cases} 0\ ,\ BX\ mod\ 2 = 0 \\ BX\ ,\ BX\ mod\ 2 = 1 \end{cases}\ \Lambda\ \ y = \begin{cases} BY\ ,\ BY\ mod\ 2 = 0 \\ 0\ ,\ BY\ mod\ 2 = 1 \end{cases}$
7 ↙	$\exists x,y[\ (x < BX\ \Lambda\ y < BY) \Longrightarrow$ $(x = x - 1,\ \ y = y + 1,\ \ n = n + 1,\ \forall i\ (M[x,y].W[i] = \theta[n].W[i])]$
8 →	$x < BX \Longrightarrow (x = x + 1,\ \ n = n + 1,\ \ \forall i\ (M[x,y].W[i] = \theta[n].W[i]))$
9 ↗	$\exists x,y[\ (y < BY\ \Lambda\ x < BX) \Longrightarrow$ $(x = x + 1,\ \ y = y - 1,\ \ n = n + 1,\ \forall i\ (M[x,y].W[i] = \theta[n].W[i])]$
10 ↓	$y < BY \Longrightarrow (y = y + 1,\ \ n = n + 1,\ \ \forall i\ (M[x,y].W[i] = \theta[n].W[i]))$
11	**Repeat** 7-8-9-10 **until** $x = BX - 1\ \ \Lambda\ \ y = BY$
12	$x = x + 1, n = n + 1, \forall i\ (M[x,y].W[i] = \theta[n].W[i])$,



where BX = BY = 4. The first loop terminates after the 14 steps by using 2nd, 3rd, 4th and 5th conditions. The second loop terminates after the 24 steps by using 7th, 8th, 9th and 10th conditions.



where BX = BY = 3. The first loop terminates after the 9 steps by using 2nd, 3rd, 4th and 5th conditions. The second loop terminates after the 15 steps by using 7th, 8th, 9th and 10th conditions.

Fig. 3. Sequential assignment of the initial weight values which come from K-Means++

After training neurons without the target attribute, all correct classes for each data are obtained. Because a neuron can contain more data than one, containing the elements of the same class or not could be showed with colored neurons. For the visual compares, the dataset with 699 vectorial tuples is used. This dataset contains 10 attributes and it is about Breast Cancer Wisconsin (BCW) [19]. There are two certain classes and each neuron which contains the elements of the same class is shown by a different color on maps.

On all maps in the following figures, there are three different colored neurons as silver, grey and black. The silver and grey neurons show the correct clustered neurons. In the other words, these neurons contain the elements of the same class; however, the black neurons contain the elements of the different classes and the number of black neurons shows the instability of the map.

4.1 Visual Compares at the Beginning Maps

Before starting to train neurons in SOM algorithm, initializing the weight values of neurons by a pre-treatment supplies a stability to the map at the beginning immediately. In the following versions, the distinction, between the sequential assignments according to the similarities of the centre points which are returned from K-Means clustering algorithms and the random assignments of them, is observed, too.

In Fig. 4, the beginning map for the standard SOM algorithm with 20x20 neurons is shown. The initial weight values are assigned randomly in the version of the standard SOM; therefore, the map is observed indecisively.

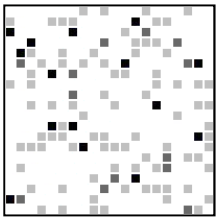
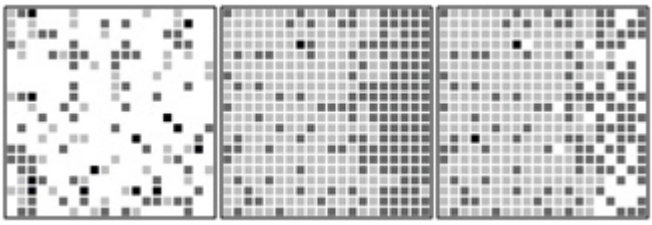


Fig. 4. Standard SOM at the beginning phase by using 20x20 neurons

On the first map in Fig. 5, there are neurons with initialized weight values by the centre points which are returned from K-Means clustering. On the second map, there are neurons with initialized weight values by the initial centre points of K-Means++ without K-Means clustering. On the third map, there are neurons with initialized weight values by the centre points which are returned from K-Means++ clustering.

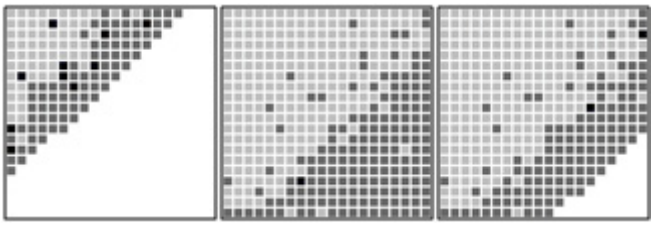


a b c

Fig. 5. SOM is at the beginning phase by using 20x20 neurons without any training a) K-Means, b) K-Means++ (without K-Means phase), c) K-Means++

These weight values are not located by using the sequential assignment and these neurons on the maps are at the beginning phase of SOM algorithm. However, it seems that the instability cannot be prevented in these examples.

In Fig. 6, the sequential assignment is used and neurons are located according to the similarities. The importance of the sequential arraignment is observed from the maps in Fig. 6.



a b c

Fig. 6. SOM is at the beginning phase by using 20x20 neurons with initialized weights by the sequential assignment. a) K-Means, b) K-Means++ (without K-Means phase), c) K-Means++ (with K-Means Clustering)

It is observed at the visual tests that the successes of the new versions of SOM algorithm have more stability and less indecisive neurons than the standard SOM. Also, visually, it can be obtained that using K-Means++ without K-Means clustering has a near success together with using K-Means++ with K-Means clustering for the sequential initialized weight values and initialized number of iteration of SOM as the number of data (699 in the previous examples).

The numerical comparisons of the new versions are done by calculating the error rates at the phase of training neurons in SOM algorithm.

4.2 Visual Compares at the Beginning Maps

The indecisive neurons are shown by black neurons on the previous maps. They mean that irrelevant data tuples are in same neurons on the map and if the number of these neurons is high, the map is not consistent.

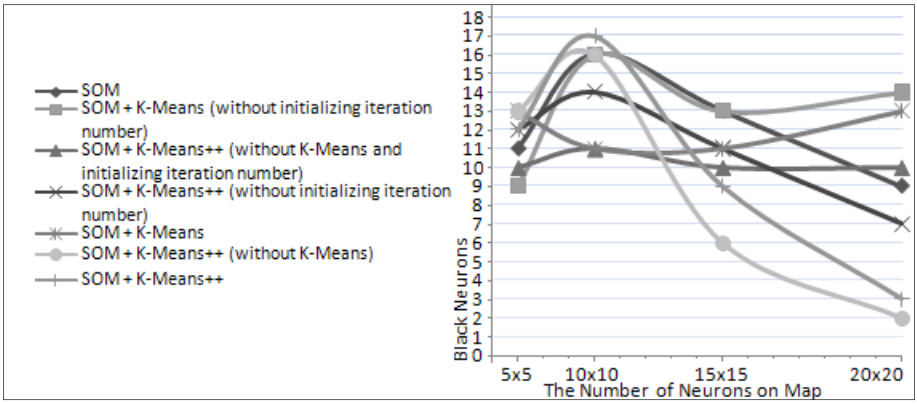


Fig. 7. The comparison of indecisive neurons

The numbers of indecisive neurons in the previous visual compares are collected like the graph in Fig. 7. This graph shows that because the numbers of total neurons for the first 5x5 neurons are low, the numbers of indecisive neurons are low, too. Therefore, the numbers of indecisive neurons are higher for 10x10 neurons and the versions of SOM could be compared according to the numbers of the indecisive neurons for 10x10, 15x15 and 20x20 neurons.

The steadiest algorithms are SOM++ (or SOM + K-Means++ (without K-Means)) and SOM + K-Means++ according to the graph. Both of these algorithms use both initializing iteration number as the number of data and the sequential assignment algorithm.

As a result, the most successful algorithm is SOM++ with the least number of indecisive neurons as 6 for 15x15 neurons and only 2 for 20x20 neurons.

4.3 The Error Rates

Error rates are calculated according to is the Gaussian Function in Eq. 3 and they are obtained for all versions of algorithms like the other comparison tests; however, Table 1 shows the least error results only for K-Means++ without K-Means clustering and with K-Means clustering.

In Table 1, a1) Error rates for K-Means++ (without K-Means) + SOM; error rates for K-Means++ (without K-Means clustering) + SOM with the initialized weights by the sequential assignment according to the number of neurons and the number of iterations b1) Error rates for K-Means++ (without K-Means)+ SOM with the initialized number of iteration as the number of total data (699) and without the initialized weights by the sequential assignment according to the number of neurons and the number of iterations. c1) Error rates for K-Means++ (with K-Means clustering) + SOM with the initialized number of iteration as the number of total data (699) and the initialized weights by the sequential assignment according to the number of neurons and the number of iterations.

Error Rates for K-Means++ (with K-Means Clustering) + SOM; a2) Error rates for K-Means++ (with K-Means clustering) + SOM with the initialized weights by the sequential assignment according to the number of neurons and the number of iterations b2) Error rates for K-Means++ (with K-Means Clustering)+ SOM with the initialized number of iteration as the number of total data (699) and without the initialized weights by the sequential assignment according to the number of neurons and the number of iterations. c2) Error rates for K-Means++ (with K-Means clustering) + SOM with the initialized number of iteration as the number of total data (699) and the initialized weights by the sequential assignment according to the number of neurons and the number of iterations.

Table 1. The error rates list for 25 and 100 neurons

	Iteration Number	5x5 neurons	10x10 neurons
a1	1th	0.42720	0.49013
	2nd	0.01000	0.01873
	3rd	0.00518	0.00177
b1	1th	1.61001	0.68055
	2nd	0.08828	0.01924
	3rd	0.03508	0.00733
c1	1th	0.28725	0.05095
	2nd	0.00252	0.00058
	3rd	0.00125	0.00007
a2	1th	0.43723	0.47991
	2nd	0.05145	0.01509
	3rd	0.01086	0.00112
b2	1th	1.81375	0.67474
	2nd	0.09418	0.01972
	3rd	0.03524	0.00777
c2	1th	0.28313	0.08267
	2nd	0.00795	0.00029
	3rd	0.00298	0.00024

4.4 Training Times

The tests of training times are implemented by using the dataset with 18781 vectorial tuples and 390 attributes. This large dataset is produced the distinctions on performance of trainings between simple SOM, SOM++ (with the initialization method of K-Means++), SOM + K-Means++ and SOM + K-Means absolutely.

The computer, which tests the performances of the algorithms for this large dataset, has 3.24 GB of RAM and Intel(R) Core(TM) 2 Duo CPU E6550 @ 2.33 GHz. This computer trains 600x600 neurons with this large dataset which has a large attribute set for three weeks.

The Fig. 8 shows that the simple SOM trains 600x600 neurons since the first moment; however, SOM++ (with the initialization method of K-Means++), SOM +

K-Means and SOM + K-Means++ need a preparation time for SOM. Therefore, the error rates of SOM are observed stably until a few times for these versions of SOM. These times are less than an hour for SOM++, about 2 hours for SOM + K-Means++ and about 3 hours for SOM + K-Means, because there are lots of attributes and tuples in the dataset. After these times, the simple SOM starts with initialized weight values and iteration values for these versions.

At the end of 7 days, the simple SOM gives about 2.4 of the error rate; however, SOM++ gives a less error rate than $10x10^{-7}$ before 8 days. Also, the error rate for SOM + K-Means++ is taken a less error rate than $10x10^{-7}$ before 9 days and the error rate for SOM+ K-Mean is taken a less error rate than $10x10^{-7}$ before 10 days. These results show that the versions of SOM have better performances than the simple SOM. However, SOM++ has the best performance. Because of the complexity of SOM algorithm as $O(N^2)$ where N is the input vector size (N is 390 x 18781 and N^2= 53649618668100), SOM algorithm gets the result map with minimum error rates after some days.

Also, it is observed that the initialization method of center points at K-Means++ accelerates K-Means, because K-Means++ with all steps needs about 2 hours to cluster a large dataset. However, K-Means needs about 3 hours.

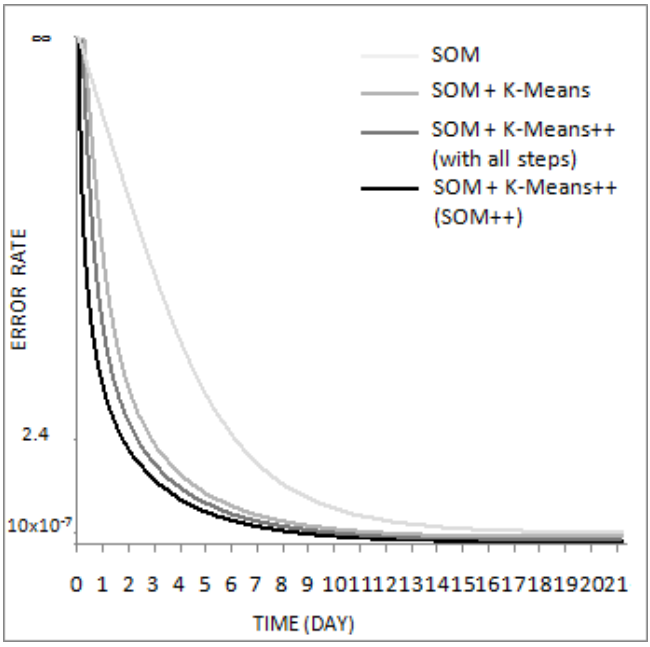


Fig. 8. The error rates - time (day) graphic for the versions of SOM

5 Comparison Results

The accuracy, about which of the SOM versions must be used, is taken by the visual tests. It is observed that the SOM versions of K-Means++ without K-Means clustering

and with K-Means clustering have the least number of indecisive neurons in the visual tests. If the versions of K-Means++ with K-Means clustering and without K-Means clustering have a close success, a comparison of time can do the distinction between them and it is observed that K-Means++ (without K-Means clustering) + SOM has the best results.

Consequently, our experimental results empirically prove that K-Means++ (without K-Means clustering) + SOM (SOM++ with its short name) is best suited to data clustering due to its high speed and lower error rates as compared with other SOM based techniques.

6 Summary, Conclusion and Future Work

This paper introduces a new clustering algorithm SOM++. The significant difference between SOM++ algorithm and the standard SOM is that SOM++ does not start to initialize the weight values of neurons with random numbers. SOM++ uses the initializing center points of clusters method in K-Means++. Eventually, each neurons represent a cluster and thus, SOM++ takes advantage of K-Means++. Separately, these significant initial values are not located in the neurons on the map and SOM++ has a special locating algorithm namely the sequential assignment.

Another difference between SOM++ algorithm and the standard SOM is that SOM++ initializes the starting number of iteration. Because the standard SOM starts with the random values in neurons, the number of iteration is declared as 0. However, because SOM++ starts with significant values in neurons, the number of iteration is declared as the number of total data. This initializing increases stability and decreases error rates.

In other words, SOM++ algorithm has many advantages over conventional SOM based methods. The most remarkable advantage of SOM++ is in saving training time for clustering large and complicated data sets by using K-Means++ algorithm in the weight initialization procedure of SOM. Furthermore, the rate of unstable data points decreases and internal error decreases.

For future work, the proposed algorithm, SOM++**,** can be used for the computer security, the healthcare, ecological modeling, the financial sector and another area which needs clustering its large data successfully.

References

1. Aguado, D., Montoya, T., Borras, L., Seco, A., Ferrer, J.: Using SOM and PCA for Analysing and Interpreting Data from a P-removal SBR. Engineering Applications of Artificial Intelligence 21(6), 919–930 (2008)
2. Arthur, D., Vassilvitskii, S.: K-Means++ the Advantages of Careful Seeding. In: Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1027–1035 (2007)
3. Attik, M., Bougrain, L., Alexandre, F.: Self-organizing map initialization. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3696, pp. 357–362. Springer, Heidelberg (2005)
4. Benabbas, F., Khadir, M.T., Fay, D., Boughrira, A.: Kohonen Map Combined to the K-Means Algorithm for the Identification of Day Types of Algerian Electricity Load. IEEE Proc. 7th Computer Information Systems and Industrial Management Applications, 78–83 (2008), doi:10.1109/CISIM.2008.27

5. Chi, S.-C., Yang, C.-C.: A Two-stage Clustering Method Combining Ant Colony SOM and K-means. Journal of Information Science and Engineering 24, 1445–1460 (2008)
6. Chi, S.-C., Yang, C.C.: Integration of Ant Colony SOM and K-Means for Clustering Analysis. In: Gabrys, B., Howlett, R.J., Jain, L.C. (eds.) KES 2006. LNCS (LNAI), vol. 4251, pp. 1–8. Springer, Heidelberg (2006)
7. Chiu, C.-Y., Chen, Y.-F., Kuo, I.-T., Ku, H.-C.: An Intelligent Market Segmentation System Using K-Means and Particle Swarm Optimization. Expert Systems with Applications 36(3), 4558–4565 (2008)
8. Chiu, C.-Y., Kuo, I.-T., Chen, P.-C.: A Market Segmentation System for Consumer Electronics Industry Using Particle Swarm Optimization and Honey Bee Mating Optimization. Global Perspective for Competitive Enterprise, Economy and Ecology pt. 12, ch. 1 (2009)
9. Corrêa, R.F., Ludermir, T.B.: A Hybrid SOM-Based Document Organization System. In: IEEE Proc. 9th Brazilian Symposium on Neural Networks (SBRN 2006), pp. 90–95 (2006), doi:10.1109/SBRN.2006.3
10. Khedairia, S., Khadir, M.T.: Self-Organizing Map and K-Means for Meteorological Day Type Identification for the Region of Annaba -Algeria-. In: IEEE Proc. 7th Computer Information Systems and Industrial Management Applications, pp. 91–96 (2008), doi:10.1109/CISIM.2008.29
11. Kohonen, T.: The Self-Organizing Map. Proc. of the IEEE 78(9), 1464–1479 (1990), doi:10.1109/5.58325
12. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proc. of 5th Berkeley Symposium on Mathematical Statistic and Probability, vol. 1, pp. 281–297. University of California Press, Berkeley (1967)
13. Poelmans, J., Elzinga, P., Viaene, S., Van Hulle, M.M., Dedene, G.: How Emergent Self Organizing Maps can Help Counter Domestic Violence. In: IEEE Proc. 2009 WRI World Congress on Computer Science and Information Engineering (CSIE), Los Angeles, USA, vol. 4, pp. 126–136 (2009), doi:10.1109/CSIE.2009.299
14. Roussinov, D.G., Chen, H.: A Scalable Self-organizing Map Algorithm for Textual Classification: A Neural Network Approach to Thesaurus Generation. Communication and Cognition in Artificial Intelligence Journal 15(1-2), 81–111 (1998)
15. Sagheer, A., El., T.N., Maeda, S., Taniguchi, R., Arita, D.: Fast Competition Approach using Self Organizing Map for Lip-Reading Applications. In: IEEE Proc. International Joint Conference on Neural Network (IJCNN), pp. 3775–3782 (2006), doi:10.1109/IJCNN.2006.1716618
16. Souza, J.R., Ludermir, T.B., Almeida, L.M.: A Two Stage Clustering Method Combining Self-Organizing Maps and Ant K-Means. In: Alippi, C., Polycarpou, M., Panayiotou, C., Ellinas, G. (eds.) ICANN 2009, Part I. LNCS, vol. 5768, pp. 485–494. Springer, Heidelberg (2009)
17. Su, M.-C., Liu, T.-K., Chang, H.T.: Improving the self-organizing feature map algorithm using an efficient initialization scheme. Tamkang Journal of Science and Engineering 5(1), 35–48 (2002)
18. Wolberg, W.H.: Breast Cancer Wisconsin (Original) Dataset (1992), `http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29`
19. Yang, Y., Rong, L.: Establishment of the Evaluation Index System of Emergency Plans Based on Hybrid of SOM Network and K-means Algorithm. In: IEEE Proc. 4th International Conference on Natural Computation, pp. 347–351 (2008), doi:10.1109/ICNC.2008.454