Centroid Neural Network for Unsupervised Competitive Learning

Dong-Chul Park

***Abstract*—An unsupervised competitive learning algorithm based on the classical k-means clustering algorithm is proposed. The proposed learning algorithm called the centroid neural network (CNN) estimates centroids of the related cluster groups in training date. This paper also explains algorithmic relationships among the CNN and some of the conventional unsupervised competitive learning algorithms including Kohonen's self-organizing map (SOM) and Kosko's differential competitive learning (DCL) algorithm. The CNN algorithm requires neither a predetermined schedule for learning coefficient nor a total number of iterations for clustering. The simulation results on clustering problems and image compression problems show that CNN converges much faster than conventional algorithms with compatible clustering quality while other algorithms may give unstable results depending on the initial values of the learning coefficient and the total number of iterations.**

***Index Terms*—Centroid, forgetting, learning gain, neural network, unsupervised learning.**

I. Introduction

CONVENTIONAL competitive learning algorithms for unsupervised learning in artificial neural networks have been widely used for processing the input data of complicated classification tasks. One of the most widely used competitive learning algorithms is the k-means clustering algorithm [1]. Since the k-means clustering algorithm that minimizes the energy function defined by mean squared error is simple and fast enough to be performed in real time, it is frequently used for many applications even though it has some inevitable problems [2]. The most serious problem with the k-means clustering algorithm is that the algorithm may not converge to an optimal solution for $k > 1$.

The self-organizing map (SOM) by Kohonen does have a strong connection with the k-means algorithm [3]. Lloyd's batch k-means algorithm [5] and MacQueen's adaptive k-means algorithm [6] are considered as the basis for the SOM [2]. Basically, SOM finds a winner neuron which is the closest to a given input datum and updates the synaptic weights of the winner and its neighbors. In order to obtain the best results from SOM, the initial learning coefficient and the total number of iteration for a given set of data should be chosen carefully. Generally, the larger the predetermined total number of iterations is and the smaller the initial learning coefficient is, the better the results that can be expected. However, it is not possible to determine *a priori* the best total number of iterations for a given set of data.

The differential competitive learning (DCL) algorithm introduces concepts of reward and punishment to the competitive learning algorithm [7]. The DCL algorithm rewards the winner by adapting the synaptic vectors with a positive learning coefficient like SOM does, but it can punish the loser by adapting the synaptic vector with a negative learning coefficient. This concept of punishment has not been used in conventional competitive learning algorithms including SOM. Even though Kohonen uses this concept for the learning vector quantization (LVQ) system, the LVQ is a supervised learning algorithm [3]. The DCL can be thought of as a local unsupervised approximation of Kohonen's supervised LVQ algorithm. However, choosing a schedule for optimal learning coefficients still remains unsolved in DCL.

The proposed learning algorithm, called centroid neural network (CNN), is based on the observation that synaptic vectors converge to the centroids of clusters as learning proceeds in conventional unsupervised competitive learning algorithms such as SOM or DCL. The centroid, or conditional expectation, can minimize the mean-squared error of the vector quantization. As is the case with SOM or DCL, the synaptic vectors converge to the centroids of clusters as learning proceeds in CNN. However, the CNN finds locally optimal synaptic vectors for each datum presented and consequently converges to the centroids of clusters much faster than conventional algorithms. One of the very advantageous features in the CNN algorithm is that the CNN does not require a schedule for learning coefficients. The CNN rather finds its optimal learning coefficient in each representation of data vectors. The CNN can also reward and punish by learning coefficients for winners and losers, respectively. Unlike SOM or DCL, the CNN also does not require the total number of iteration in advance.

II. Definition of Problem

Assume that N data vectors, $\boldsymbol{x}(i), i = 1, \cdots, N$, and number of clusters, M, are given. Each of the datum is grouped as a member of one of the M clusters. When the cluster i has N_i members and $\boldsymbol{x}_i(j)$ denotes the data j in the cluster i for one instance, the problem and the energy function (or cost function), E, in L_2 norm are defined as follows:

$$\text{Find } \{\boldsymbol{w}_i,\ 1 \le i \le M\}$$

$$\text{such that minimize } E = \sum_{i=1}^{M} E_i = \sum_{i=1}^{M} \sum_{j=1}^{N_i} ||\boldsymbol{x}_i(j) - \boldsymbol{w}_i||^2$$

$$\text{with } N = \sum_{i=1}^{M} N_i$$

Manuscript received June 22, 1998; revised February 2, 1999 and October 4, 1999. This work was supported by the Development Program for the Exemplary Schools in Information and Communications from the Ministry of Information and Communication (MIC) of Korea.

The author is with the Intelligent Computing Research Lab., School of Electrical and Information Control Engineering, Myong Ji University, Yong In, Kuung Ki-do 449–728, Korea (e-mail: parkd@wh.myongji.ac.kr).

Publisher Item Identifier S 1045-9227(00)02998-2.

where $\boldsymbol{w}$ denotes a weight vector that has the same dimension as $\boldsymbol{x}$.

III. Optimality Conditions in Unsupervised Learning Algorithm

In order to achieve the minimum-energy clustering, the following conditions are considered [9].

1) *Nearest Neighbor Selection Condition*: The "winner" neuron i for a given input $\boldsymbol{x}$ is defined as

$$\boldsymbol{w}_i = \min_j \|\boldsymbol{x} - \boldsymbol{w}_j\|^2, \qquad 1 \le j \le M.$$

2) *Minimum Energy Condition*: The weights for a given output neuron, should be chosen in a way to minimize the total distance in L_2 norm from the vectors in its cluster such as

$$\boldsymbol{w}_i = \min_{\boldsymbol{w}} \sum_{j=1}^{N_i} \|\boldsymbol{x}_i(j) - \boldsymbol{w}\|^2 = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{x}_i(j)$$

by the following Theorem 1.

Theorem 1: The centroid of data in a cluster is the solution which gives the minimum energy in L_2 norm.

IV. Some Algorithms on Unsupervised Competitive Learning

In order to attain insight into learning gain scheduling in different unsupervised competitive learning algorithms, some conventional algorithms are summarized in this section.

A. Deterministic Competitive Learning

One of the simplest forms of the deterministic competitive learning algorithm is the following the linear competitive learning algorithm [8]:

$$\boldsymbol{w}_j[n+1] = \boldsymbol{w}_j[n] + S_j(y_j)(\boldsymbol{x} - \boldsymbol{w}j[n])$$

where

$$S_j(y_i) = \begin{cases} 1, & \text{if neuron } j \text{ is winner} \\ 0, & \text{otherwise.} \end{cases}$$

This learning algorithm states that only the weights of the winner neuron have a chance to adapt. The learning rate in this algorithm is constant through the learning stage and is desirable when the environment is nonstationary [2].

B. Self Learning Algorithm

A similar learning algorithm to the deterministic competitive learning algorithm is the following self learning algorithm [9]:

$$\boldsymbol{w}_j[n+1] = \boldsymbol{w}_j[n] + \frac{1}{N_j[n]}(\boldsymbol{x} - \boldsymbol{w}_j[n])$$

where $N_j[n]$ is the number of training data which output neuron j won at iteration n.

This update rule is rather closely related with the MacQueen's k-means algorithm [6] and guarantees that the cluster centers are the means of all input patterns which have been assigned to them. Unfortunately, this algorithm does not guarantee the quality of the solution or the rate of convergence especially for $k > 1$ where k represents the number of clusters [2].

C. Self-Organizing Map

Kohenen's self-organizing map (SOM) as an unsupervised learning algorithm is one of the most popular neural-network algorithms and produces several variations for improving the performance of SOM. The SOM can be summarized as

$$\boldsymbol{w}_j[n+1] = \boldsymbol{w}_j[n] + \alpha[n](\boldsymbol{x} - \boldsymbol{w}_j[n])$$

where output neuron j is the winner neuron and neighbor neurons to the winner j at the iteration n and the learning coefficient $\alpha[n]$ at iteration n defines a decaying constant with an iteration such as

$$\alpha[n] = c(0)\left(1 - \frac{n}{N}\right)$$

with predetermined constant $c(0)$ and total number of iterations N [3], [4].

The SOM holds the very advantageous topology preserving property that can capture the probability distribution density of the input data without help of external supervision [11], [12]. As mentioned in literature including [13], however, the required parameter selection prior to learning is very important in achieving useful results. Although a recent report on the optimal scheduling of learning gain using the Kalman filter estimation technique is a very useful tool for running SOM [13], the selection of proper N and the width of the neighborhood function is left in the state of art until now.

D. Differential Competitive Learning

The differential competitive learning (DCL) proposed by Kong and Kosko [7], [10] combines two classical learning algorithms: competitive learning and differential Hebbian learning. The basic idea of DCL is that the winner neuron learns only if the status of the output neuron has been changed compared to the previous iteration. DCL updates weights by the following procedure:

$$\boldsymbol{w}_j[n+1] = \begin{cases} \boldsymbol{w}_j[n] + \alpha c[n](\boldsymbol{x} - \boldsymbol{w}_j[n]), \\ \quad \text{if neuron } j \text{ is winner} \\ \boldsymbol{w}_j[n], \\ \quad \text{if neuron } j \text{ is loser} \end{cases}$$

where

$$\alpha = \begin{cases} +1, & Y_j \text{ gets stronger} \\ -1, & Y_j \text{ gets weaker} \\ 0, & Y_j \text{ remains the same} \end{cases}$$

and $c[n]$ is a slowly decreasing sequence of small learning coefficients ($c[0] < 1$) while Y_j denotes the neuron j's competition signal in the competition layer.

The significance of DCL is that DCL allows punishment or forgetting by using negative learning gain in unsupervised learning. Kohonen employs this idea of punishment only in supervised learning [3].

TABLE I
THE CENTROID NEURAL-NETWORK ALGORITHM

```
Algorithm CNN(M,N)          [ M: number of clusters, N: number of data vectors ]
   [Initialize weights w1 and w2 ]
   Find the centroid, c, of all data vector
   Initialize w1 and w2 around c with small ε :
   w1 := c + ε,  w2 := c − ε
   N1 := 0,  N2 := 0
   k := 2,  epoch := 0
   for (k ≤ M)
      do
         loser := 0
         for ( n ≤ N)
            Apply a data vector x(n) to the network
            Find the winner neuron, j, in this epoch for 1 ≤ j ≤ k
            if (epoch ≠ 0), then Set i is winner neuron, i, for x(n) in previous epoch.
            if i ≠ j, then neuron, i, is loser neuron.
            if (epoch == 0 or i ≠ j)
               Run UpdateCNNWeight(x(n), wi, wj, epoch)
               loser := loser + 1
            endif
            n := n + 1
         endfor                          [check for all data ]
         epoch := epoch + 1
      while loser ≠ 0
      if k ≠ M
         Split group with the most error, j, by adding small vector, ε, nearby group j :
         wj = min_i Ei = min_i Σ_{k=1}^{Ni} ||xi(k) − wi||^2,  1 ≤ i ≤ M
         wk+1 := wj + ε
      endif
      k := k + 1, Nk := 0
   endfor
end


Procedure UpdateCNNWeight(x, wi, wj, epoch)
   Update winner neuron :  wj(n) := wj(n) + 1/(Nj+1) [x(n) − wj(n)]
   if (epoch ≠ 0)          [ loser neuron occurred only when epoch ≠ 0 ]
      Update loser neuron :  wi(n) := wi(n) − 1/(Ni−1) [x(n) − wi(n)]
   endif
end
```

V. THE CNN ALGORITHM

The CNN algorithm is based on the conventional k-means algorithm and finds the centroid of data in corresponding clusters at each presentation of data vectors. Instead of calculating the centroids of the clustered data for every presentation of data, the CNN algorithm updates their weights only when the status of the output neuron for presenting data has changed: that is, the weights of a winner neuron in the current epoch for the data change only when the winner neuron did not win the data in the previous presentation and the weights of the winner neuron in the previous epoch for the data change only when the neuron does not win the data in the current epoch. We call the former one "winner neuron" and the latter one "loser neuron."

Note that "epoch" as used in this paper is defined as "one presentation of whole data vectors in the data set," while "iteration" is defined as "one presentation of any data vector." For example, if we have N data vectors and P epochs of training have been performed, then there were PN iterations of training have been performed.

When a data vector $\boldsymbol{x}$ is applied to the network at time n, the adaptive equations for winner neuron i and loser neuron j in CNN can be written as follows:

$$\begin{aligned}\boldsymbol{w}_j(n+1) &= \frac{1}{N_j+1}[N_j\boldsymbol{w}_j(n)+\boldsymbol{x}(n)] \\ &= \boldsymbol{w}_j(n)+\frac{1}{N_j+1}[\boldsymbol{x}(n)-\boldsymbol{w}_j(n)] \end{aligned} \tag{1}$$

$$\begin{aligned}\boldsymbol{w}_i(n+1) &= \frac{1}{N_i-1}[N_i\boldsymbol{w}_i(n)-\boldsymbol{x}(n)] \\ &= \boldsymbol{w}_i(n)-\frac{1}{N_i-1}[\boldsymbol{x}(n)-\boldsymbol{w}_i(n)] \end{aligned} \tag{2}$$

TABLE II
COMPARISON IN LEARNING COEFFICIENTS AMONG DIFFERENT LEARNING ALGORITHMS

Algorithm	$c(n)$	Note
SLA	$\frac{1}{N_j(n)}$	N_j : the number of members in neuron j
SOM	$c(0)(1-\frac{n}{N})$	$c(0)$: arbitrary constant
DCL	$\alpha c(0)(1-\frac{t}{T})$ $\alpha = \begin{cases} (+1) & \text{if } Y_j \text{ gets stronger} \\ (-1) & \text{if } Y_j \text{ gets weaker} \\ 0 & \text{if } Y_j \text{ remains same} \end{cases}$	t: data sequence, T: total number of data Y_j : competitive signal of winner
CNN	$c(n) = \begin{cases} (+1)\frac{1}{N_j+1} & \text{if } j \text{ is winner} \\ (-1)\frac{1}{N_j-1} & \text{if } j \text{ is loser} \\ 0 & \text{otherwise} \end{cases}$	$N_j \neq 1$

TABLE III
SUMMARY OF LEARNING GAINS FOR DIFFERENT ALGORITHMS

Algorithm	Learning gain schedule	Forgetting allowed
SLA	linearly decreasing	No
SOM	linearly decreasing	No
DCL	linearly decreasing	Yes
CNN	locally optimal	Yes

where $\boldsymbol{w}_j(n)$ and $\boldsymbol{w}(i)$ represent the weight vectors of the winner neuron and the loser neuron, respectively.

In order to avoid the algorithm from getting stuck at a undesirable local minimum solution, CNN starts with setting the number of groups to two and increases the number of groups one by one until it reaches the predetermined number of groups, M. Table I shows a pseudocode of the CNN algorithm.

The CNN algorithm does not provide any guarantee of convergence to the global minimum solution like other unsupervised algorithms such as SLA, DCL, and SOM, but does provide a guarantee of convergence to a local minimum. The convergence of the CNN algorithm can be easily proven by considering the energy change in each iteration of the training procedure.

VI. RELATIONSHIP OF CNN WITH SOME UNSUPERVISED COMPETITIVE LEARNING ALGORITHMS

From (1) and (2), the adaptation rules of the CNN can be combined such that

$$\boldsymbol{w}_j(n+1) = \boldsymbol{w}_j(n) + c(n)[\boldsymbol{x}(n) - \boldsymbol{w}_j(n)] \tag{3}$$

where

$$c(n) = \begin{cases} (+1)\dfrac{1}{N_j+1}, & \text{if } j \text{ is a winner} \\ (-1)\dfrac{1}{N_j-1}, & \text{if } j \text{ is a loser and} \\ & \text{previous winner for } \boldsymbol{x}(n) \\ 0, & \text{otherwise.} \end{cases}$$

Some of the conventional unsupervised learning algorithms to be compared with CNN include SLA, DCL, and SOM. The learning coefficient schedules for these algorithms are summarized in Table II. Note that SOM does not adapt only the weights of the winner neuron j but also adapts the weights of the neighbor neurons to the winner neuron j. One of the most widely used neighborhoods is a sphere that covers a fairly large region initially and shrinks its diameter with time [4], [12], [13].

When CNN is compared with SLA, SOM, and DCL, CNN and DCL allow a forgetting factor among some of the output neurons by using negative learning gain while SLA and SOM do not. In most cases, the forgetting factor is very natural and effective in unsupervised learning [10]. The magnitude of the learning coefficient in the DCL, however, follows the idea of SLA and SOM whose learning rates are linearly decreasing with each iteration without considering the property of data and learning conditions. A simple linearly decreasing learning coefficient, of course, does not satisfy the optimality conditions given in Section III. The CNN, however, adapts its weights according to the optimality conditions. Of course, this does not mean any convergence of CNN to the global minimum. This simply shows some relationships of CNN to other algorithms. By employing the strategy of starting the group numbers at two and increasing it until it reaches to the prespecified number of groups, the CNN can improve its convergence and quality of

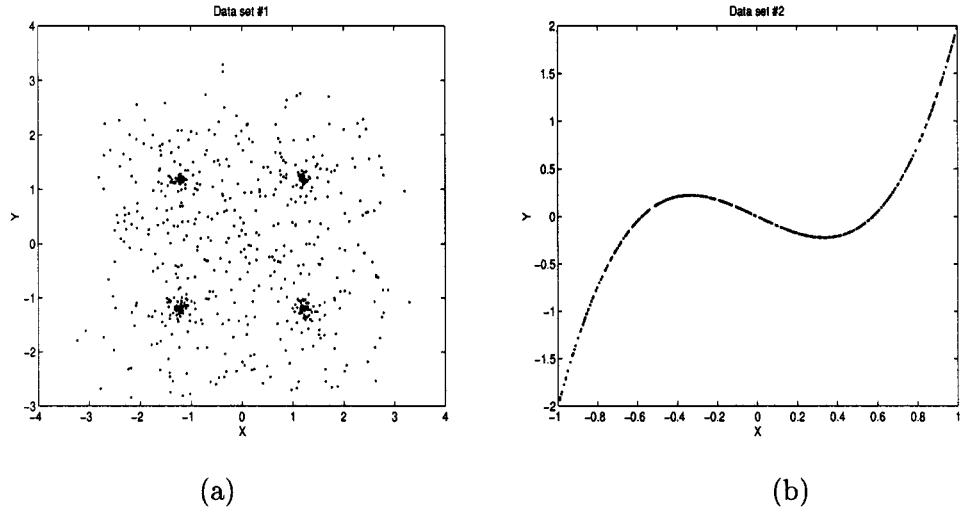

Fig. 1. Data sets used for experiment: (a) data set #1 and (b) data set #2.

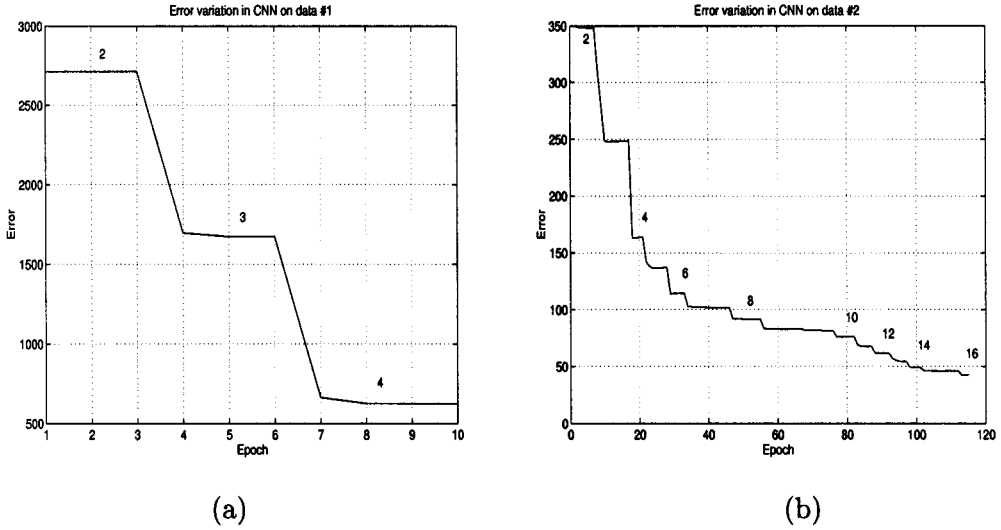

Fig. 2. Convergence of CNN: (a) for data set #1 (# of groups is four) and (b) for data set #2 (# of groups is 16). (Note: The numbers on the curve represent the number of groups while running the CNN algorithm.)

solution greatly. Table III summarizes the schedules of learning gains and forgetting properties among different algorithms.

VII. Experiments and Results

A. Example Problems

The CNN algorithm is first compared with the SOM and the DCL algorithms for example problems shown in Fig. 1. Fig. 1(a) shows the data set #1 that consists of 2000 data vectors with four centers at the four corners of -2 and $+2$ of the X–Y axis. For data set #1, the algorithms are to group the data into four clusters. Fig. 1(b) shows the data set #2 that has 500 data points with an S-curve shape in two-dimensional (2-D) space. The data set #2 is used to compare the three algorithms when the data do not show any obvious clustering. Note that the following neighborhood function, $H(n)$, for SOM is used throughout the experiments in this section

$$H(n) = \exp\left(\frac{-\tau}{\sigma(n)}\right)$$

where r is the radius from the winner neuron to the neighborhood neurons and the $\sigma(n)$, the width of the neighborhood, is chosen to cover 25% of the whole neuron initially and to decay slowly with n.

The clustering results are given in terms of the energy defined in Section II. In order to investigate the effects of the total number (N) of epochs and initial learning gain, $c(0)$, the experiments for SOM and DCL are performed with five different N's (10, 50, 100, 150, 200) and five different $c(0)$'s (0.1, 0.3, 0.5, 0.7, 0.9). Of course, CNN is performed only once for each data set since CNN requires neither N nor $c(0)$. The same initial

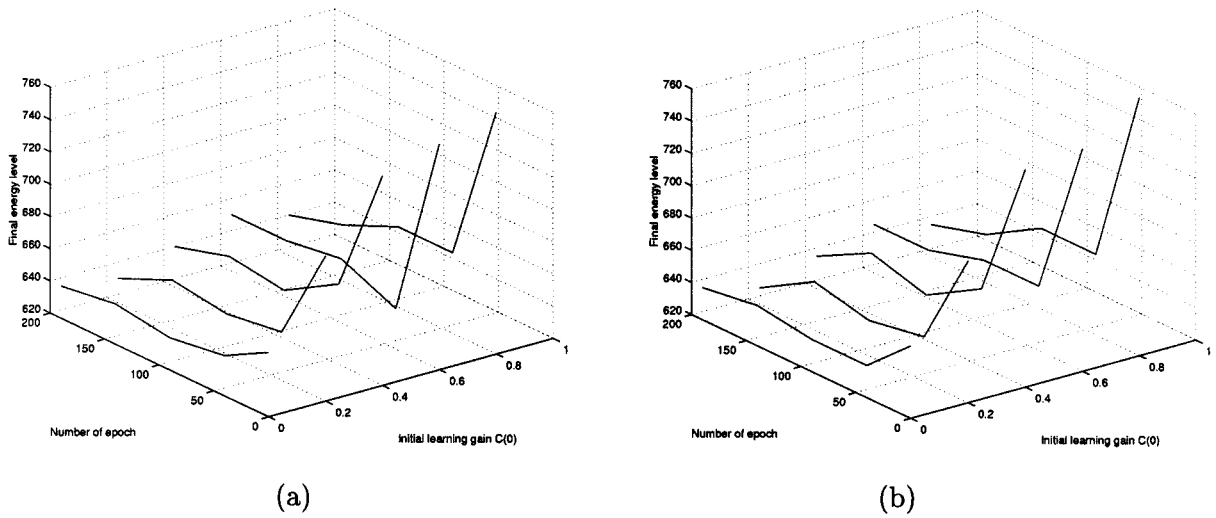

Fig. 3. Different final energy levels with different initial parameters on data set #1: (a) for SOM and (b) for DCL. (Note: CNN always gives $E = 623$ after eight epochs.)

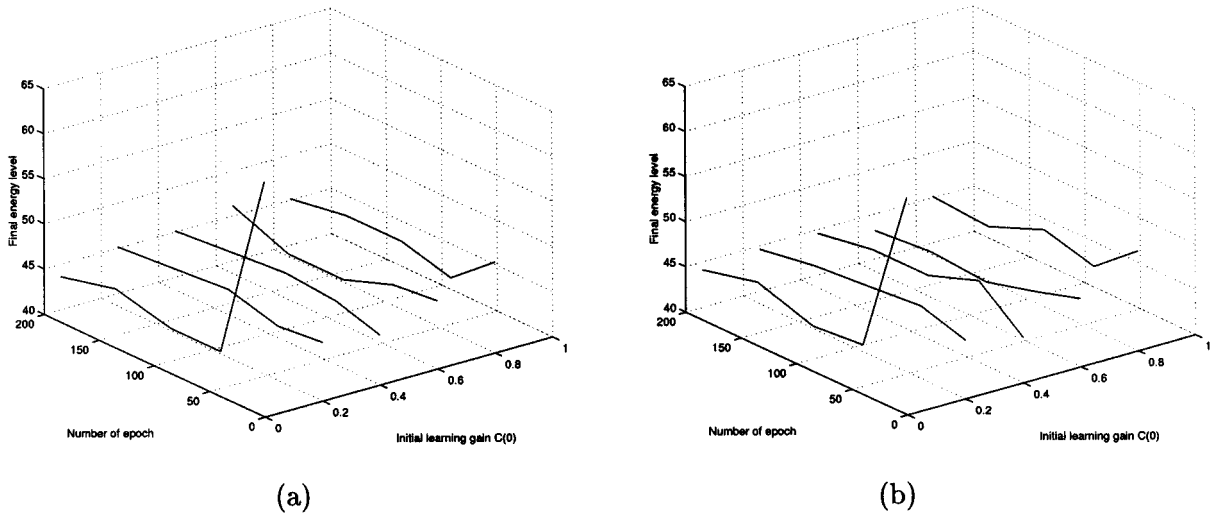

Fig. 4. Different final energy levels with different initial parameters on data set #2: (a) for SOM and (b) for DCL. (Note: CNN gives $E = 42$ after 115 epoch.)

weights around the middle of the data plane are set for the three algorithms. For both data sets, all the algorithms successfully converge to the centroids of possible groups of data. The differences among the algorithms are the speed of convergence and final energy levels. Fig. 2 shows the convergence of the CNN algorithm when applying the datasets #1 and #2 with the number of clusters 4 and 16, respectively.

The results shown in Fig. 3(a) and (b) and display the final energy level when SOM and DCL are applied to the data set #1, respectively. As we can see from Fig. 3, generally, the larger N is and the smaller $c(0)$ is, the better the results that can be expected for SOM and DCL. The best result for SOM is obtained ($E = 618$) with $N = 200$ and $c(0) = 0.3$ while the best result for DCL is $E = 617$ with $N = 200$ and $c(0) = 0.5$ among different combinations of N and $c(0)$. Both SOM and DCL show fairly good results when N is over 150, while CNN gives a comparable result ($E = 623$) after eight epochs of training. Note that the actual CPU times for one epoch in SOM and DCL algorithms used are almost the same with marginal differences. However, since the CNN has two output neurons initially and M output neurons in the final stage of training process, the CNN requires approximately half of the CPU time required for one epoch in SOM or DCL in the average sense.

The results in Fig. 4 show the final energy levels for SOM and DCL with data set #2. Since the number of data for this case is relatively small, the final energy is not very distinguishable for different cases.

The next example problem investigated with the proposed CNN algorithm is the uniform data in the 2-D plane. The 10 000 data points are uniformly distributed over the region enclosed by the boxed area and the task is to group the data into 25 clusters. The SOM and the proposed CNN are compared on this task. The $c(0)$ and N for SOM are given by 0.2 and 100, respectively. Of course, CNN requires no initial parameters to be set. Initial weights are clustered around the center of the plot. At each epoch of the training stage, the presentation order of the input data was randomized for both algorithms in order to generalize the results. With both algorithms, weights gradually

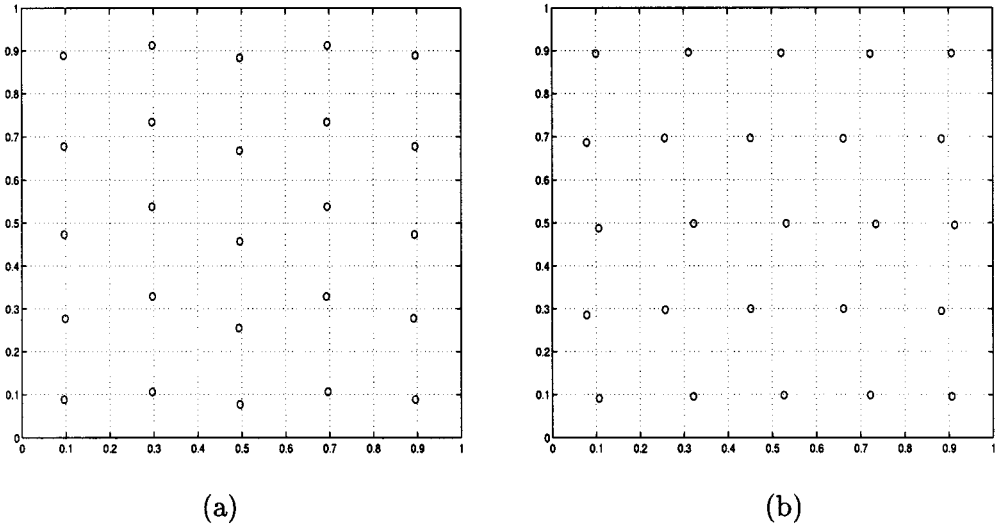

Fig. 5. Results of convergence: (a) by SOM and (b) by CNN for 10 000 uniformly distributed data points. Each point represents the two weights of each output neuron. Final energy levels for SOM and CNN are 0.006 611 and 0.006 621, respectively.

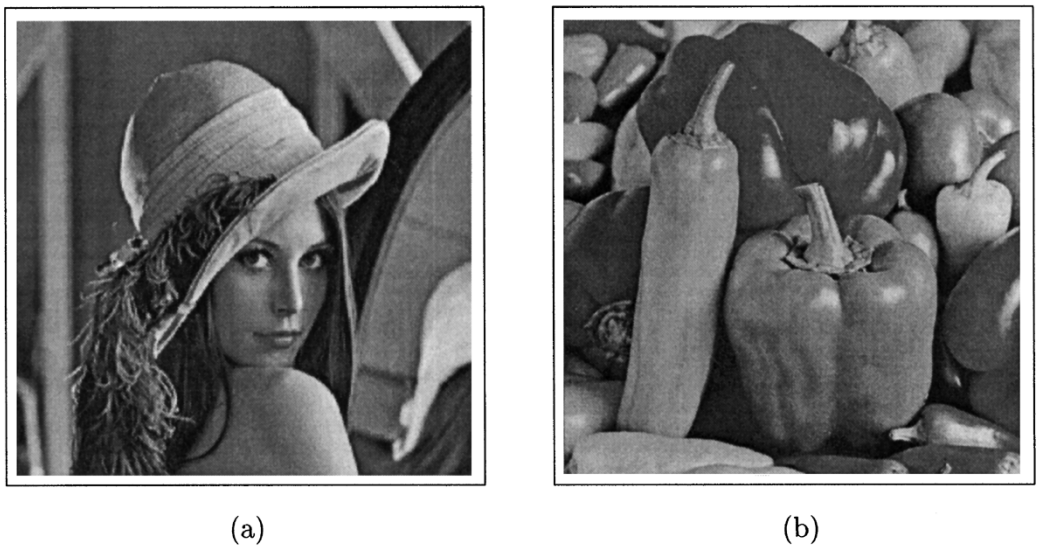

Fig. 6. The original images: (a) LENA and (b) PEPPERS.

spread out until the weight distribution approximates the uniform distribution. Fig. 5 shows the final location of weights for SOM and CNN. The results shown in Fig. 5 are quite similar with minor differences in final energy levels.

An interactive CNN simulator with several problems is available at the following web site: http://icrl.myongji.ac.kr/NNsimulator/CNN.

B. Image Compression Problem

In order to investigate the applicability of the CNN algorithm to real-world problems, the CNN is applied to image compression problems [14], [15]. When the block size is 3 $\times$ 3, the original image data, LENA (255 $\times$ 255) as shown in Fig. 6(a), consists of 7225 blocks of data. The problem requires that the 7255 blocks of data be coded with 128 blocks of coded data. Fig. 7 shows the results of image compression by the CNN. The CNN training for codebook design takes 1873 s of CPU time in SUN SPARC-20. The CNN is compared with SOM in terms of peak signal-to-noise ratio (PSNR) for the image compression problem in Table IV. Note that SOM with a different combination of parameters, $\alpha(0)$ and N, shows different results while CNN gives acceptable results without problems in parameter selection for running the algorithm.

Another experiment on image compression is performed by using CNN, SOM, and DCL. In this experiment, 512, 1024, and 2048 codebook size cases with 4 $\times$ 4 block size are considered.

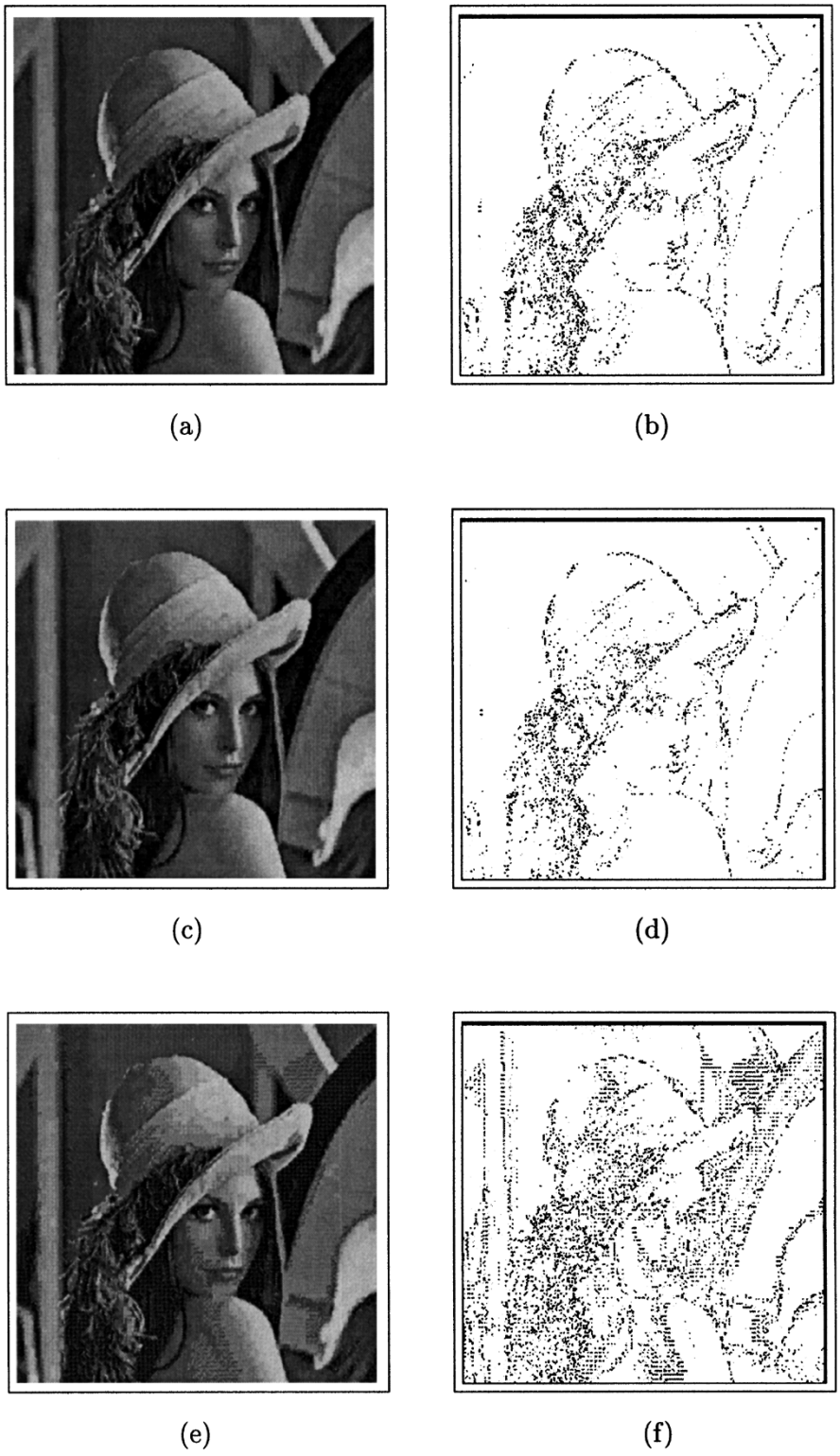

Fig. 7. Results of image compression with different initial parameters for SOM and CNN: (a) and (b) reconstructed image by CNN and its error image: SNR $= 29.26$ dB (After 400 epoch), (c) and (d) reconstructed image by SOM ($c(0)$ = 0.2 and $N = 1000$) and its error image: SNR = 29.30 dB, and (e) and (f) reconstructed image by SOM ($c(0) = 0.7$ and $N = 500$) and its error image: SNR $= 26.29$ dB.

For SOM and DCL, $c(0) = 0.2$ and $N = 1000$ epoch are used since it gives the best results in the previous problem. The PEPPERS image shown in Fig. 6(b) is used for training neural networks and resulting weights are tested with the LENA image. Table V shows the resulting PSNR comparison among the algorithms. The CPU times required for CNN, SOM, DCL for this task are given in Table VI. As can be seen from the CPU time table, CNN requires much less CPU time than what SOM and

TABLE IV
PSNR COMPARISON BETWEEN SOM AND CNN WITH DIFFERENT SETS OF TOTAL NUMBER OF TRAINING EPOCHS AND LEARNING COEFFICIENTS FOR LENA IMAGE COMPRESSION PROBLEM

Network	Total Epochs (N)	$c(0)$ 0.2	0.3	0.5	0.7
SOM	100	29.25	28.96	27.89	26.45
	300	29.29	28.94	28.16	27.06
	500	29.29	28.95	27.07	26.29
	1,000	29.30	28.88	28.10	27.37
CNN	400	29.26			

TABLE V
PSNR COMPARISON FOR LENA IMAGE COMPRESSION PROBLEM WITH DIFFERENT SIZES OF CODEBOOKS DESIGNED BY PEPPERS IMAGE

Codebook Size ($b.p.p.$)	SOM	CNN	DCL
512 (0.5625)	29.94	30.22	29.98
1,024 (0.6250)	30.43	30.32	30.39
2,048 (0.6875)	30.69	30.71	30.74

TABLE VI
CPU TIME (IN SECONDS) WITH SUN SPARC-20 REQUIRED FOR THE EXPERIMENT

Codebook Size	SOM	CNN	DCL
512	14,102	8,326	10,254
1,024	28,204	15,228	20,369
2,048	56,408	32,627	39,767

DCL use. As can be seen from Tables V and VI, CNN gives very comparable results with much less computational effort.

VIII. CONCLUSION

The CNN algorithm based on the k-means clustering algorithm is proposed. The proposed CNN algorithm has a strong connection with some of the conventional unsupervised learning algorithms. In order to obtain lower energy, the CNN dynamically allocates the synaptic weights to the clusters with high energy. Even though the CNN algorithm is not new when we consider the k-means algorithm, the CNN provides us with a new interpretation of the k-means algorithm and the relationships with some of the conventional algorithms. While applying the CNN algorithm to several problems, the CNN successfully converges to suboptimal solutions. Any undesirable local minimum problem was not observed in our CNN experiments performed with many different data sets. The CNN algorithm is applied to several problems such as simple 2-D data problems and image compression problems. When compared with conventional clustering algorithms such as Kohonen's self-organizing map and Kosko's differential competitive learning on these problems, the proposed CNN algorithm produces comparable results with less computational effort and is free of the optimum parameter selection problem.

ACKNOWLEDGMENT

The author would like to thank T. Dinh, Y. J. Woo, and Prof. R. Zarub for their help in preparing this manuscript and the anonymous reviewers for their careful reviews and constructive comments.

REFERENCES

[1] J. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
[2] C. Darken and J. Moody, "Fast adaptive k-means clustering: Some empirical results," in *Proc. 1990 Int. Joint Conf. Neural Networks*, vol. 2, 1990, pp. 233–238.
[3] T. Kohonen, *Self-Organization and Associative Memory*, 3rd ed. Berlin, Germany: Springer-Velag, 1989.
[4] ——, "The self-organizing map," *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.
[5] S. Lloyd, "Least-square quantization in PCM," Bell Labs Tech. Rep..
[6] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probability*, 1967, pp. 281–297.
[7] S. Kong and B. Kosko, "Differential competitive learning for centroid estimation and phoneme recognition," *IEEE Trans. Neural Networks*, vol. 2, pp. 118–124, Jan. 1991.
[8] S. Grossberg, "On learning and energy–entropy dependence in recurrent and nonrecurrent signed networks," *J. Statist. Phys.*, vol. 1, pp. 319–350, 1969.
[9] Y. Tsypkin, *Foundations of the Theory of Learning Systems*. New York: Academic, 1973.
[10] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamic Systems Approach to Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[11] T. Villmann *et al.*, "Topology preservation in self-organizing feature maps," *IEEE Trans. Neural Networks*, vol. 8, pp. 256–266, Mar. 1997.
[12] J. Zurada, *Int. to Artificial Neural Systems*. St. Paul, MN: West, 1992.
[13] K. Haese, "Self-organizing feature maps with self-adjusted learning parameters," *IEEE Trans. Neural Networks*, vol. 9, Nov. 1998.
[14] O. Chen, B. Sheu, and W. Fang, "Image compression using self-organization networks," *IEEE Trans. Circuits, Syst., Video Technol.*, vol. 4, pp. 480–489, Oct. 1994.
[15] C. Amerijckx *et al.*, "Image compression by self-organized Kohonen maps," *IEEE Trans. Neural Networks*, vol. 9, pp. 503–507, May 1998.
[16] *IEEE Trans. Inform. Theory*, vol. IT-28, p. 129, 1982.