Branching Competitive Learning Network: A Novel Self-Creating Model

Huilin Xiong, M. N. S. Swamy, *Fellow, IEEE*, M. Omair Ahmad, *Fellow, IEEE*, and Irwin King, *Member, IEEE*

***Abstract*—This paper presents a new self-creating model of a neural network in which a branching mechanism is incorporated with competitive learning. Unlike other self-creating models, the proposed scheme, called branching competitive learning (BCL), adopts a special node-splitting criterion, which is based mainly on the geometrical measurements of the movement of the synaptic vectors in the weight space. Compared with other self-creating and nonself-creating competitive learning models, the BCL network is more efficient to capture the spatial distribution of the input data and, therefore, tends to give better clustering or quantization results. We demonstrate the ability of the BCL model to appropriately estimate the cluster number in a data distribution, show its adaptability to nonstationary data inputs and, moreover, present a scheme leading to a multiresolution data clustering. Extensive experiments on vector quantization of image compression are given to illustrate the effectiveness of the BCL algorithm.**

***Index Terms*—Competitive learning, self-creating neural network, vector quantization.**

I. Introduction

COMPETITIVE learning neural networks have been developed for applications such as data clustering and vector quantization [1]–[6]. Compared to the conventional clustering algorithms, e.g., the K-mean algorithm, sometimes known as the generalized Lloyd algorithm (GLA) [7]–[9], the algorithms based on competitive learning of neural networks offer the advantages of online operation and require very little storage. However, like the K-mean algorithm, most competitive learning neural networks, such as the *self-organizing feature map* (SOFM) [10] and *frequency sensitive competitive learning* (FSCL) [1], [11], need to assume a network with a fixed number of nodes, which means that the cluster number of the input data set must be pre-specified in advance. In the situation where there is no *a priori* information available about the underlying data distribution, it is very difficult to appropriately estimate the cluster number in a data set. As a result, it is often realized only at the end of the experiment that a different cluster number setting might be more appropriate.

Manuscript received April 3, 2002; revised January 14, 2003 and September 15, 2003. This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada, by the Fonds la Formation des Chercheurs et l'Aideà la Recherche (FCAR), Québec, Canada, and by a Grant from the Research Grants Council of the Hong Kong Special Administration Region, China.

H. Xiong, M. N. S. Swamy, and M. O. Ahmad are with the Center for Signal Processing and Communications, Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: hlxiong@ece.concordia.ca; swamy@ece.concordia.ca; omair@ece.concordia.ca).

I. King is with the Multimedia Information Processing Laboratory, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: king@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TNN.2004.824248

An attractive way to deal with this problem is including a self-creating mechanism in the competitive learning process so that the network can adaptively determine its size. Some self-creating models have been proposed in the literature [12]–[14]. *Self-creating and organizing neural network* (SCONN) [12] employs an adaptively modified error threshold to control the network's self growth. Fritzke's *growing cell structures* (GCS) [13] assigns each node with a local variable, which can dynamically increase or decrease during the competitive learning. After a period of learning, the node with the largest value of the local variable gets the right to generate a new node. Based on the GCS model, Fritzke has proposed the *growing neural gas* (GNG) model [15], [16], which employs a more delicate criterion to insert and delete nodes. It has been shown in [15] that the GNG and its modification are good at learning topologies and the recently modified GNG, the *growing neural gas with utility criterion* (GNG_U) [16], is able to track nonstationary data input.

In this paper, we present a new self-creating model of neural network, called the branching competitive learning (BCL) network. Unlike other self-creating models, e.g., the SCONN, GCS, and GNG, the proposed model employs a special node-splitting criterion, based mainly on the geometrical measurements of the movement of the synaptic vectors in the weight space. The basic idea behind adopting the geometrical measurements is that the learning tracks of the synaptic vectors in the weight space can help in probing the spatial distribution of the input data, from which the network knows as to when or where to split its nodes is appropriate. In the BCL, each of the nodes from its birth is assigned a local value, called activation level, to measure the intensity of its corresponding synaptic vector's oscillatory behavior in the weight space. For a data presentation, if the measurements of the winner's movement surpass the pre-specified thresholds, this activation is viewed as a valid activation, and an increment is added to the winner's activation level; otherwise, not only the activation levels of the losers, but also that of the winner is decreased with a "forgetting" factor λ. As a node's activation level becomes large enough, a new node is generated from it, whereas, the nodes with low winning chance are deleted. In this way, the BCL network, starting from one node in the output layer, dynamically splits and prunes its nodes along with the competitive learning and, finally, presents a good clustering result with an appropriate number of nodes. Compared with the other self-creating models, the BCL network is more sophisticated to capture the spatial distribution of the input data and, therefore, usually provides a better clustering or quantization results. Besides, in the BCL algorithm, the feature of being able to

easily control the resolution can make it simple to implement a multiresolution data clustering.

Simulations are carried out to show the ability of the BCL to appropriately estimate the cluster number in a data distribution, the adaptability of the BCL for handling nonstationary data input, and how the BCL can give rise to a multiresolution data clustering. Moreover, to test the effectiveness of the BCL model and compare it with other models, extensive experiments of vector quantization for image data compression are conducted by using the BCL, GLA and other learning vector quantization algorithms.

The paper is organized as follows. In Section II, we review some conventional competitive learning models, including the self-creating and nonself-creating models. In Section III, we describe in detail the branching competitive learning model and the BCL algorithm. In Section IV, we present some simulation results on synthetic data sets to illustrate the performance of the BCL for estimating the cluster number in a data set, dealing with the so-called *stability-plasticity dilemma* [12], and clustering data in a multiresolution fashion. Experiments on vector quantization design and the comparison results are given in Section V. Finally, Section VI concludes the paper by highlighting the contribution of this study.

II. Data Clustering and Competitive Learning Networks

Assume there are N data vectors in d-dimensional space, $\{\vec{x}_i\}_{i=1}^{N}$ and $\vec{x}_i \in \mathbf{R}^d$, and the cluster number M is pre-specified. Then, the process of data clustering can be defined as follows:

Find $\{\vec{\omega}_i\}_{i=1}^{M}$ in $\mathbf{R}^d$ to minimize the average distortion or the mean squared error (MSE) given by

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N_i}\|\vec{x}_i(j) - \vec{\omega}_i\|^2 \tag{1}$$

where N_i is the number of data in cluster i, $\vec{x}_i(j)$ represents the data point in cluster i ($j = 1, 2, \ldots, N_i$), $N = \sum_{i=1}^{M} N_i$, and $\|\cdot\|$ is the L_2 norm.

Competitive learning networks are known to be effective for the task of data clustering [17]. The simple competitive learning (SCL) model is a winner-take-all scheme, whose algorithm can be described as

$$\begin{aligned}\vec{\omega}_c(t+1) &= \vec{\omega}_c(t) + \alpha_c(\vec{x} - \vec{\omega}_c(t))\\ &\text{for } c = \arg\min_j \|\vec{x} - \vec{\omega}_j\|^2\end{aligned} \tag{2}$$

where $\vec{x}$ is a randomly selected input data point, t represents the current step of competitive learning, $\vec{\omega}_j$ denotes the synaptic vectors corresponding to the jth neural unit (for simplicity, $\vec{\omega}_j$ also is used to represent the jth neural unit or node), and α_c is the learning rate. In practice, to guarantee the convergence of the learning procedure, a gradually decreasing learning rate is often adopted

$$\alpha_c(t) = \alpha_0\left(1 - \frac{t}{T}\right) \tag{3}$$

where α_0 is the initial learning rate, and T denotes a pre-specified number of iterations.

The simple competitive learning algorithm is very easy to implement. However, it suffers from the so called *dead unit* problem, which means that some nodes may never be activated by the competition. To deal with this problem, some modified schemes have been developed. Kohonen's SOFM [10] employs a winner-take-quota strategy to alleviate the *dead unit* problem. In SOFM, not only the winner, but also the winner's neighbors can learn from the competition

$$\begin{aligned}\vec{\omega}_c(t+1) &= \vec{\omega}_c(t) + \alpha_c(\vec{x} - \vec{\omega}_c(t))\\ &\text{for } c = \arg\min_j \|\vec{x} - \vec{\omega}_j\|^2\\ \vec{\omega}_b(t+1) &= \vec{\omega}_b(t) + \alpha_b(\vec{x} - \vec{\omega}_b(t))\\ &\text{for all } b \in N_c(t) \quad b \neq c\end{aligned} \tag{4}$$

where N_c denotes the neighborhood centered at the winner c and α_b is the neighbors' learning rate. Usually, $\alpha_b \leq \alpha_c$, and N_c, α_b and α_c all decrease along with the competitive learning procedure. Although SOFM can alleviate the *dead unit* problem to some extent, it cannot eliminate the dead unit completely. Besides, its performance is greatly affected by the selection of the neighborhood function; ill-adjusted neighborhoods may lead SOFM to perform poorly.

Another modification of SCL is FSCL [1], [11]. The competitive learning of the FSCL is different from that of the SCL simply in that a winning frequency term is added to avoid the situation that some units always fail in the competition

$$\begin{aligned}\vec{\omega}_c(t+1) &= \vec{\omega}_c(t) + \alpha_c(\vec{x} - \vec{\omega}_c(t))\\ &\text{for } c = \arg\min_j \gamma_j\|\vec{x} - \vec{\omega}_j\|^2\end{aligned} \tag{5}$$

where γ_j is the frequency that $\vec{\omega}_j$ has won the competition up to current competition step; that is $\gamma_j = n_j / \sum_i^t n_i$ and n_i is the cumulative number of $\vec{\omega}_i$ winning the competition. Obviously, when the winning frequency of a unit becomes large enough, the chance that it continues to win will become small. On the contrary, as the winning rate of a unit becomes small enough, the probability of it winning the next competition becomes large. In this way, the FSCL achieves a nearly equal node utilization and, therefore, avoids the dead units. However, many experiments show that the competitive learning aiming at equally or "fairly" utilizing each node cannot generally result in a better data clustering or vector quantization than GLA and SCL.

The above competitive learning algorithms have one thing in common: the network size M must be pre-specified; that is, the cluster number M for the input data set need to be pre-specified. Because there is usually no *a priori* information available to appropriately estimate the cluster number, we have to try different values of M to obtain a better clustering result. An attractive way to deal with this problem is to add a self-creating mechanism to the neural network so that the network can automatically increase its size to an appropriate level according to the input data distribution.

There are some self-creating models in the literature [12]–[15]. The self-creating and organizing neural network (SCONN) [12] employs an adaptively modified error threshold

to control the network's self-growth. It is shown in [12] that SCONN could represent the data structure more efficiently than the SOFM. Another important self-creating model is Fritzke's GCS [13], which can be viewed as a modification of SOFM by including a node-insertion mechanism. Each node in GCS is assigned with a local variable called "signal counter." For a data presentation, following the winner-take-quota procedure that the winner and its neighbors learn from the competition, the winner's signal counter τ_c increases with a constant increment s as $\tau_c(t+1) = \tau_c(t) + s$ and then, the signal counters of all the nodes decay with a "forgetting" factor α as $\Delta\tau_j = -\alpha\tau_j$, where $j = 1, \ldots, M(t)$, $M(t)$ denoting the node number at time t, and $0 < \alpha < 1$. After a fixed number of data presentations λ ($\lambda = 100$, as recommended in [13]), a new node is inserted between the node with the largest value of the signal counter and its farthest neighbor. In GCS, network occasionally needs to delete the so-called *superfluous* node, whose synaptic vector is located in a region with a very low probability density. For the sake of simplicity, nodes with signal counter being less than a specified threshold or nodes that never win during a complete iteration are regarded as the *superfluous* nodes and removed. Besides the GCS model, Fritzke also proposed some state-of-the-art self-creating models such as the GNG [15] and GNG_U [16], which employ more delicate criteria to insert and delete nodes.

III. Branching Competitive Learning Network

We approach the self-creating mechanism with a different criterion which is based on two geometrical measurements of the synaptic vectors' competitive movements in the weight space. Some basic ideas can be found in a previous work [18].

A. The Branching Criterion and the BCL Algorithm

Intuitively, in the process of competitive learning, when a synaptic vector exhibits an intensively oscillatory movement in the weight space, it usually means that the vector is "attracted" by two or more different data clusters. At this moment, splitting the synaptic vector is appropriate to decrease the clustering distortion substantially. Geometrically, the larger the change in the moving direction of a winner and the longer its movements in two consecutive activations, the more intense is the winner's oscillation. Thus, in the weight space, as shown in Fig. 1, the angle between two consecutive moving directions, i.e., $\mathrm{ang}(\vec{x}(t_c) - \vec{\omega}_c,\ \vec{x}(t_l) - \vec{\omega}_c)$ and the minimum value of two consecutive moving distances, i.e., $\min(\|\vec{x}(t_c) - \vec{\omega}_c\|,\ \|\vec{x}(t_l) - \vec{\omega}_c\|)$ can be used to measure the intensity of a winner's oscillatory behavior.

In our BCL scheme, each node (or synaptic vector) is associated with a local variable, called activation level, to estimate the intensity of the synaptic vector's oscillatory behavior. When a node is activated by a competition (or wins a competition), we check if this is a valid activation meaning that the two geometrical measurements of the winner's movement in the weight space surpass the pre-specified thresholds

$$\begin{cases} \mathrm{ang}(\vec{x}(t_c) - \vec{\omega}_c,\ \vec{x}(t_l) - \vec{\omega}_c) > \varphi_0 \\ \min(\|\vec{x}(t_c) - \vec{\omega}_c\|,\ \|\vec{x}(t_l) - \vec{\omega}_c\|) > d_0 \end{cases} \tag{6}$$

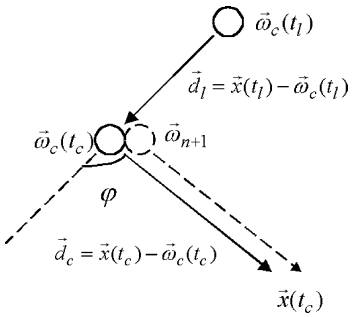


Fig. 1. An illustration of a valid activation, where $\varphi > \varphi_0$ and $\min(\|\vec{d}_l\|, \|\vec{d}_c\|) > d_0$.

where $\vec{\omega}_c$ denotes a winner in the current competition, $\vec{x}(t_c)$ and $\vec{x}(t_l)$ represent the two data presentations in two consecutive activations of $\vec{\omega}_c$, t_c denotes the current competition step, t_l represents the previous activated step, and φ_0 and d_0 denote the angle and distance thresholds pre-specified to control the branching process. Usually, the threshold φ_0 in (6) is set to 90° and, therefore, the first condition of a valid activation becomes

$$(\vec{x}(t_c) - \vec{\omega}_c) \cdot (\vec{x}(t_l) - \vec{\omega}_c) < 0. \tag{7}$$

For a data presentation $\vec{x}(t)$, assume that $\vec{\omega}_c$ is the winner of the current competition. Let

$$\mu^{\vec{\omega}_j}(t) = \begin{cases} 1 & \text{when } \vec{x}(t) \text{ leads to a valid activation of } \vec{\omega}_j \\ 0 & \text{otherwise} \end{cases}$$

where $j = 1, 2, \ldots M(t)$, and $M(t)$ is the node number at time t. The activation level of $\vec{\omega}_j$ at the current step is updated by

$$L^{\vec{\omega}_j}(t) = \begin{cases} L^{\vec{\omega}_j}(t-1) + \Delta L & \mu^{\vec{\omega}_j}(t) = 1 \\ (1-\lambda) L^{\vec{\omega}_j}(t-1) & \mu^{\vec{\omega}_j}(t) = 0 \end{cases}$$

where ΔL is a pre-specified constant increment of the activation level and λ denotes a "forgetting" factor $0 < \lambda < 1$. When the value of a winner's activation becomes large enough, i.e., greater than a pre-specified threshold, $L^{\vec{\omega}_c}(t) > L_0$, a new node is generated from $\vec{\omega}_c$ as

$$\vec{\omega}_{M(t)+1} = \vec{\omega}_c + \alpha(t)(\vec{x}(t) - \vec{\omega}_c).$$

Meanwhile, the activation levels of the "mother" and "son" are set as $L^{\vec{\omega}_c}(t) = 0$ and $L^{\vec{\omega}_{M(t)+1}}(t) = 0$.

In practice, the BCL may sporadically generate a few dead nodes that are no longer or seldom activated by the competitive learning. To avoid this situation, we simply employ a threshold β called pruning rate, to delete the dead nodes: if $\gamma_j < \beta$, then delete $\vec{\omega}_j$ from the set of nodes, where γ_j denotes the frequency that $\vec{\omega}_j$ has won the competition from its birth to the current competition step. We now formulate the BCL algorithm in Fig. 2.

In step 6 of the BCL algorithm, "dynamical equilibrium" means that the fluctuation of the network size remains relatively small for a long period of learning. We can use a simple criterion such as the one given in Section V-B to decide if the network size has reached its dynamical equilibrium point. Although the BCL network with a reasonable parameter setting always reaches the dynamical equilibrium, in some classical

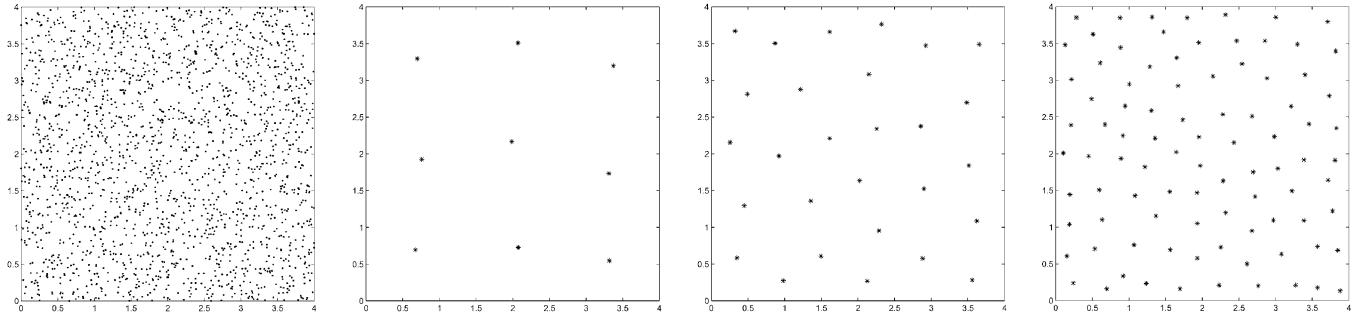
Branching Competitive Learning (BCL) Algorithm

1. Initialize the "seed" or the first synaptic vector, and set its activation level to 0.
2. Randomly take a sample $\vec{x}(t)$ from the data set, find the winner $\vec{w}_c$ of current competition in the set of synaptic vector $\{\vec{w}_j\}$ $(j = 1, 2, \cdots, M(t))$, and decrease all except the winner's activation level as $L^{\vec{w}_j}(t) = (1-\lambda)L^{\vec{w}_j}(t-1)$, where $j \neq c$.
3. If the measurements of $\vec{w}_c$ satisfies (6), its activation level increases by $\triangle L$; otherwise, decrease the activation level by $-\lambda L^{\vec{w}_c}(t-1)$.
4. If $L^{\vec{w}_c}(t) > L_0$, generate a new node $\vec{w}_{M(t)+1}$ from $\vec{w}_c$ as

$$\vec{w}_{M(t)+1} = \vec{w}_c + \alpha_c(\vec{x} - \vec{w}_c),$$

and set $L^{\vec{w}_c}(t) = 0$ and $L^{\vec{w}_{M(t)+1}}(t) = 0$; otherwise, just update $\vec{w}_c$ by $\vec{w}_c + \alpha_c(\vec{x} - \vec{w}_c)$.
5. For each unit $\vec{w}_j$, if its winning frequency γ_j from its birth to the current competition step is less than the pre-specified pruning rate β, i.e., $\gamma_j < \beta$, then delete the node $\vec{w}_j$ from the network.
6. If the network size reaches a dynamical equilibrium or a pre-specified number, the network stops growing and follows the simple competitive learning; otherwise, go back to step 2.

Fig. 2. The BCL algorithm.



(a) (b) (c) (d)

Fig. 3. A uniform data set and the quantization results using the BCL algorithm for different values of d_0. (a) A uniform data set. (b) Quantization result with $d_0 = 1.0$. (b) Quantization result with $d_0 = 0.5$. (d) Quantization result with $d_0 = 0.25$.

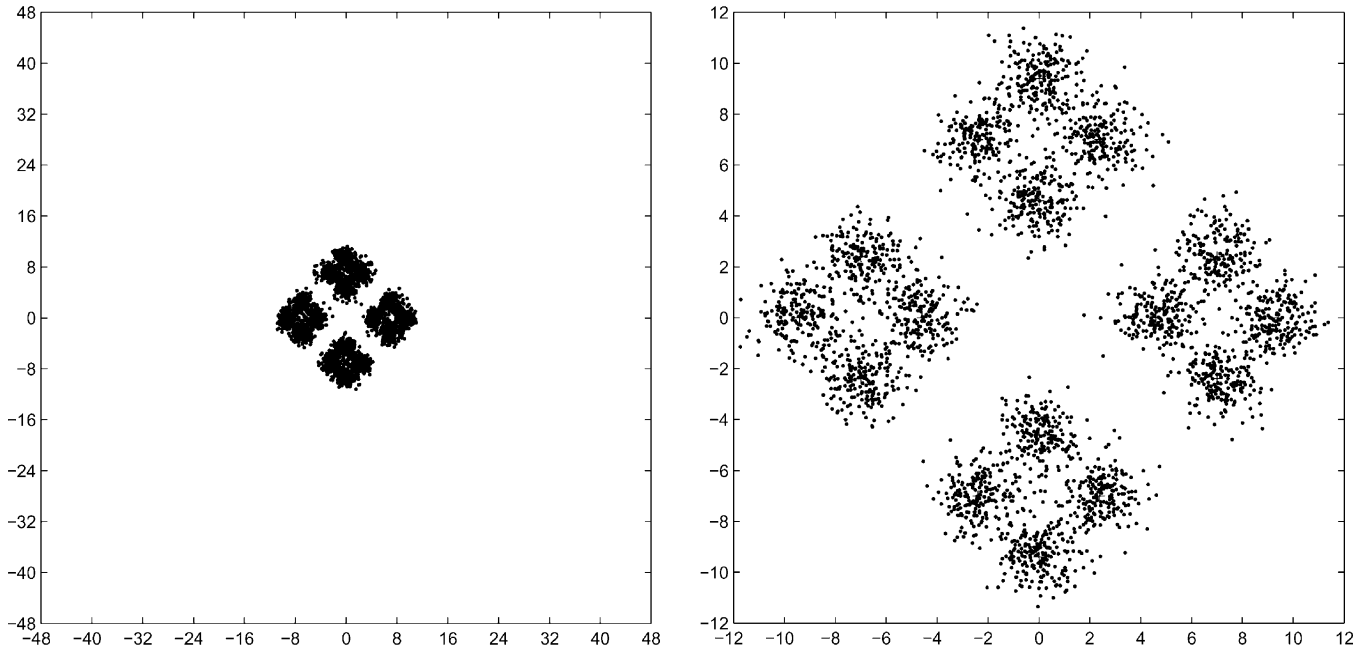
applications such as vector quantization for image coding, the codebook size, i.e., the network size, is usually pre-specified. In this case, we do not need to find the dynamical equilibrium point. Once the network size reaches the pre-specified number, we simply let the network stop growing, and then the network just follows the simple competitive learning procedure. So, the entire BCL algorithm can be divided into two phases: growing phase and data clustering phase. In the first phase, the network keeps growing until its size reaches a dynamical equilibrium or a pre-specified value. Following this, the network will simply modify the synaptic vectors to approximate the cluster centroids and no further split occurs.

B. Parameter d_0 and Multiresolution Clustering

There are seven parameters in the above BCL algorithm: α_c, φ_0, d_0, ΔL, λ, L_0, and β. Optimization of these parameters is a difficult or even an impossible task, since the optimization of some of the parameters is related to the input data distribution. However, experimentally, the BCL algorithm is not sensitive to parameter tuning. We usually set $\varphi_0 = 90°$, $\Delta L = 1.0$, and the "forgetting" factor λ is often adopted in the form of $1/N$, where N denotes the sample number in the data set.

Among the parameters, the distance threshold d_0 is of great importance to control the branching process. Generally speaking, d_0 represents the resolution level at which the BCL partitions the data set. It can also be viewed as the error bound that we can tolerate when we say two data samples belong to the same data cluster. With a large value of d_0, the BCL network will give a clustering result under a coarse resolution. On the other hand, a relatively small value of d_0 means that the BCL network "views" the data set under a fine resolution. This feature of being able to easily control the resolution is very useful for applications such as the multiresolution data compression and multiscale data visualization. Fig. 3(a) shows 2000 uniform data in square field [0,4] $\times$ [0.4] and Fig. 3(b)–(d), respectively show the quantization results by using the BCL network with $d_0 = 1.0$, 0.5, and 0.25.

From a multiresolution point of view, the estimation of the cluster number in a data set is resolution-dependent and, moreover, data clustering can also be viewed as a problem of multiresolution spatial partition. For example, from the viewpoint of a coarse resolution [Fig. 4(a)], we can say that the data set just has four clusters. However, when we view the same data set in a relatively fine resolution [Fig. 4(b)], we find four small clusters in each cluster. Hence, the cluster number along with the data clustering itself is resolution-dependent. In the BCL model, the parameter d_0 represents the resolution level at which the BCL clusters or "views" the data set. With different values of d_0, it is reasonable that we obtain different estimations of the cluster number (see Fig. 3).



(a) Coarse resolution (b) Fine resolution

Fig. 4. A multiresolution data set.

There are many hierarchical techniques in the literature [20] to implement a tree-structured data clustering. However, most of them involve intensive computation. In the BCL scheme, using an appropriately decreasing sequence of d_0, we can easily implement a multiresolution data clustering in a hierarchical structure, as given below.

1. Select a sequence of d_i $(i = 1, 2, \ldots, m)$.
2. Under the resolution level d_1, implement BCL clustering on data set D, which clusters D into n_1 subsets: $D_1^{(1)}, D_2^{(1)}, \ldots, D_{n_1}^{(1)}$, $D = \bigcup D_i^{(1)}$ and $D_i^{(1)} \bigcap D_j^{(1)} = \emptyset$, $i \neq j$.
3. Under the resolution level d_2, repeat the clustering in each subset $D_i^{(1)}$. Continue the process under all the resolution levels.

IV. Simulations on Synthetic Data Sets

In this section, we conduct three sets of experiments on synthetic data sets in order to: 1) examine the ability of the BCL to appropriately estimate the cluster number in a data set according to the given resolution level; 2) show the validity of the multiresolution data clustering by the BCL network; and 3) compare the adaptability of the BCL with other competitive learning models in handling nonstantionary input.

In practice, apart from the situation that the final cluster number is pre-specified, we usually consider the BCL network to have reached its dynamical equilibrium when the cluster number remains unchanged for quite a long period of time, e.g., $10N$ (10 iterations), where N denotes the total number of samples in the input data set.

A. Cluster Number Estimation

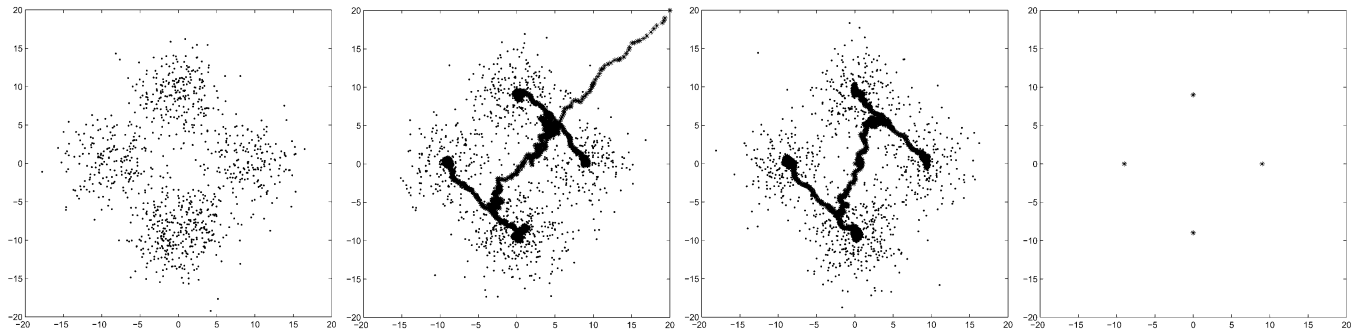
In the first simulation study, we use two sets of two-dimensional data, as shown in Figs. 5(a) and 6(a). Data set 1 contains four clusters of Gaussian data with $\sigma = 3$, and the four clusters have 200, 250, 200, and 350 samples, respectively. Obviously, there is some overlapping between the clusters. Fig. 5(b) and (c) shows the learning and branching traces of the synaptic vectors under the "seed" initialized by the point [20,20] [Fig. 5(b)], or by the origin [Fig. 5(c)]. Data set 2 consists of five Gaussian clusters ($\sigma = 3$) and the five clusters contain 200, 250, 200, 350, and 500 samples, respectively. Like data set 1, we can see that there is some overlapping between these clusters. The learning and branching traces are shown in Fig. 6(b) with the "seed" initialized by the point [25,25], and in Fig. 6(c) with the "seed" initialized by the origin.

In the experiments, we fix the learning rate with $\alpha_c = 0.02$. The threshold of the angle and the distance in the branching criterion are set as $\varphi_0 = 90^\circ$ and $d_0 = 3\sigma = 9.0$; the other parameters are set as $\Delta L = 1.0$, $L_0 = 10.0$, and $\beta = 1/3$. We adopt the "forgetting" factor in the form of $\lambda = 1/N$, where N denotes the sample number in the data set. From the learning traces, it can be seen that the BCL can appropriately estimate the cluster number in the data distribution, even in the case when there are some overlapping between the clusters.

B. Multiresolution Data Clustering

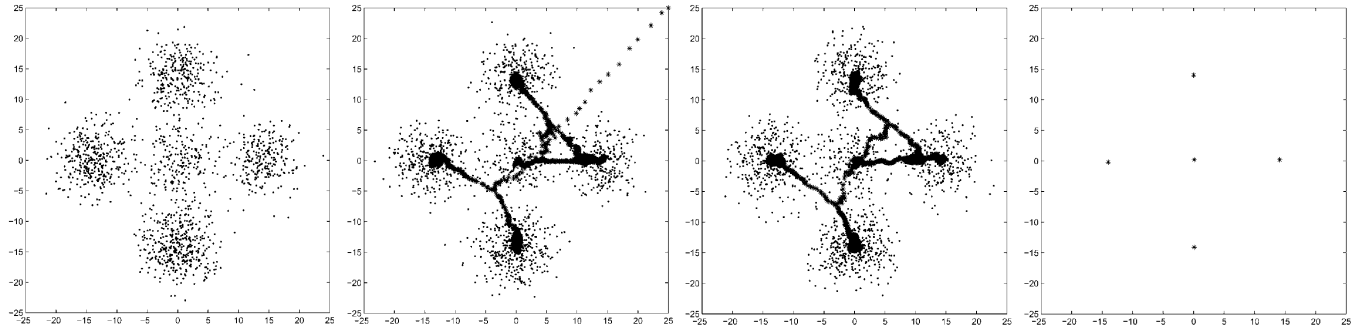
In the second simulation study, the data set, as shown in Fig. 7, consists of 16 Gaussian distributions with standard variance $\sigma = 0.8$. Each cluster contains 200 samples, and the total number of samples is 3200.

A two-level multiresolution clustering is conducted. In level 1, we set the distance threshold as $d_1 = 6\sigma = 4.8$, while in



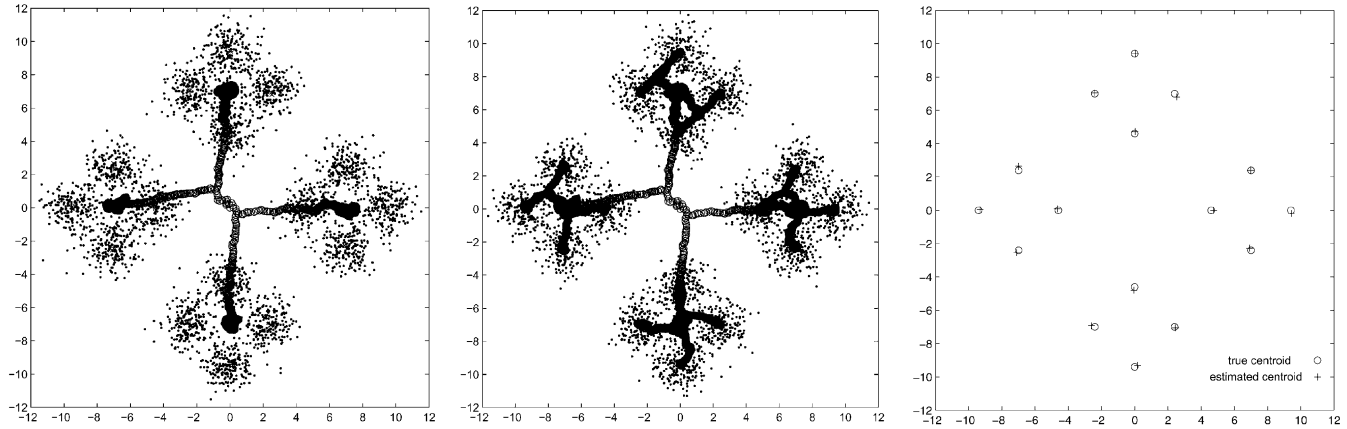
(a) (b) (c) (d)

Fig. 5. Data set 1 and the learning branching traces of the synaptic vectors. (a) Data set. (b) Learning trace from (20,20). (c) Learning trace from (0,0). (d) Final estimated centroids of (c).



(a) (b) (c) (d)

Fig. 6. Data set 2 and the learning branching traces of the synaptic vectors. (a) Data set. (b) Learning trace from (25,25). (c) Learning trace from (0,0). (d) Final estimated centroids of (c).



(a) (b) (c)

Fig. 7. A two-level multiresolution clustering by using the BCL network. (a) Resolution level 1. (b) Resolution level 2. (c) The estimated centroids and true centroids.

level 2 the distance threshold is set as $d_2 = 3\sigma = 2.4$. The other parameters are set to the same values in both the levels: $\Delta L = 1.0$, $L_0 = 10.0$, and $\beta = 1/3$. The learning rate is fixed with $\alpha = 0.02$, and the "seed" or the original synaptic vector is initialized by the point (0,0).

The results are shown in Fig. 7. Fig. 7(a) shows the learning traces of the synaptic vectors in level 1. We can see that in the resolution level 1, the data points are divided into four clusters and the convergent synaptic vectors are correctly located at the centroids of each clusters. Fig. 7(b) shows the learning traces of the further branching in the resolution level 2. We can see that the final estimated cluster number has an exact value of 16, the actual cluster number. Fig. 7(c) shows the difference between the estimated centroids and the actual data centroids.

C. Handling Nonstationary Data Input

Most competitive learning models suffer from the so-called *stability-plasticity dilemma* [12], [14], i.e., the synaptic vectors

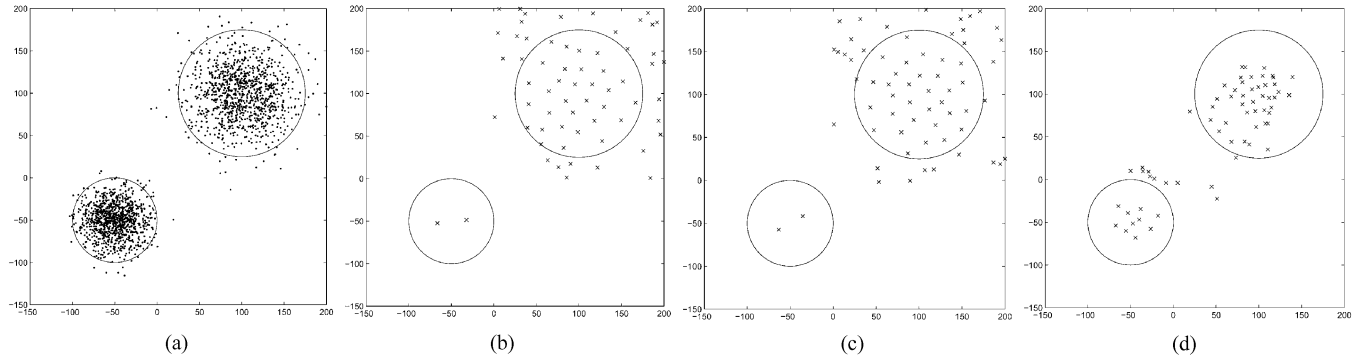


Fig. 8. (a) Nonstationary data input and the vector quantization results by using (b) SCL, (c) SOFM, and (d) FSCL, respectively.

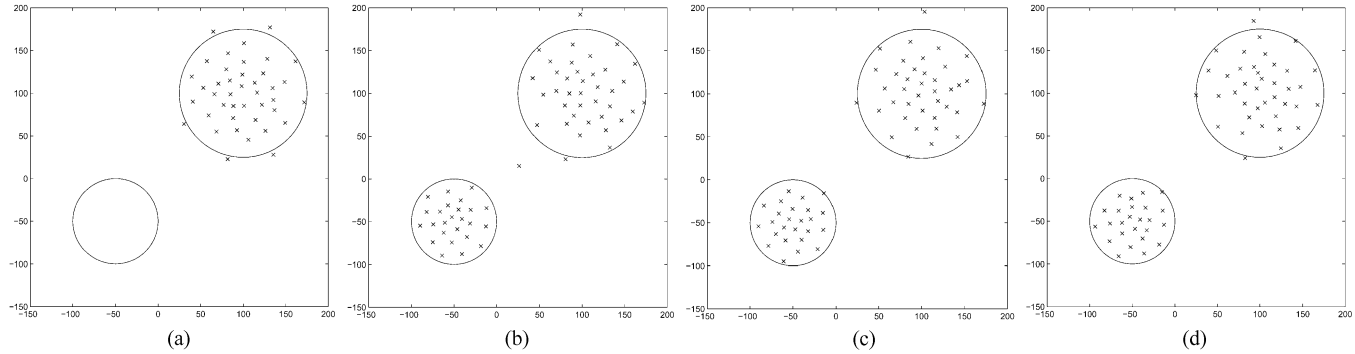


Fig. 9. Handling nonstationary input by the GNG, after (a) 20 000, (b) 40 000, (c) 80 000, and (d) 90 000 data presentations. The network size (or the cluster number) at the end of the data presentation is (a) $S_c = 40$, (b) $S_c = 64$, (c) $S_c = 64$, or (d) $S_c = 64$, respectively.

cannot adapt to the case of nonstationary data input. To demonstrate the difference between the BCL model and other competitive learning models in dealing with a nonstationary input, we conduct a third simulation on the data set as in [14] and [21]. In order to simulate a nonstationary input, for $t < 20\,000$, we let all the input data come from a Gaussian data set with $\mu_x = \mu_y = 100, \sigma_x = \sigma_y = 30$; however, for $t \geq 20\,000$, all the input data are randomly selected from another Gaussian data distribution with the parameters as $\mu_x = \mu_y = -50$, $\sigma_x = \sigma_y = 20$. The two data distributions are shown in Fig. 8(a). Each of these contains 1000 samples.

To avoid the node's "freezing" caused by the gradually decreasing learning rate given by (3), we fix the value of the learning rate at 0.02, and the synaptic vectors are first initialized by a set of randomly selected points in a square field [0,200]×[0,200]. Fig. 8(b)–(d), respectively, shows the learning quantization results after 40 000 data presentations, 20 iterations, by using the SCL, SOFM, and FSCL models (cluster number or codebook size is pre-specified as 64). We can see that, besides many dead nodes, there are very few nodes that can successfully move to the new data distribution after $\mathrm{t} = 20\,000$ for the SCL and SOFM models. As for the FSCL, although the situation is improved considerablely, its performance in dealing with nonstationary data input is still unacceptable.

We examine the performance of the GNG model in dealing with the nonstationary input problem. The parameters of the GNG model (see [15] for details) are set as: $\lambda = 500$, $\epsilon_b = 0.02$, $\epsilon_n = 0.0006$, $a_{\max} = 120$, $\alpha = 0.5$, and $d = 0.9995$. The"age" increment of all the "edges" emanating from the winner in each competition is set to 1.0. Fig. 9(a)–(d), respectively, shows the quantization results of the GNG network after 20 000, 40 000, 80 000, and 90 000 data presentations, where the symbol S_c denotes the network size (or codebook size) at the end of the data presentation. For the BCL model, the first synaptic vector is initialized by a random point in the square [0,200]×[0,200], and the learning rate is also fixed at 0.02. The other parameters are set as $d_0 = 9.0$, $L_0 = 10.0$, $\Delta L = 1.0$, $\beta = 1/3$, and $\lambda = 1/N$, where N denotes the sample number in the data set. Fig. 10(a)–(d), respectively, show the experimental results after 20 000, 40 000, 80 000, and 90 000 data presentations. In the simulations for these two models, the maximum network size is set to 64.

It can be seen that both the GNG and BCL models can adapt to the input distribution change, but the nodes of the GNG model located in the upper-circle region remain almost unchanged after 40 000 data presentations. On the contrary, due to the pruning mechanism in the BCL, the nodes in the BCL model can continue adapting after 40 000 data presentations. Hence, compared with the GNG model, the BCL model is more adaptive in dealing with the nonstationary input.

We also compare the BCL model with a recently modified GNG model, namely, the GNG_U [16], in handling nonstationary inputs. For this purpose, we implement the GNG_U algorithm using the same parameter values as in [16]: $\lambda = 500$, $\epsilon_b = 0.05$, $\epsilon_n = 0.0006$, $\beta = 0.0005$ (that means $a, d = 0.9995$), $a_{\max} = 120$, and $k = 3.0$. Figs. 11 and 12 show the simulation results (see [16] for details about the input data set).

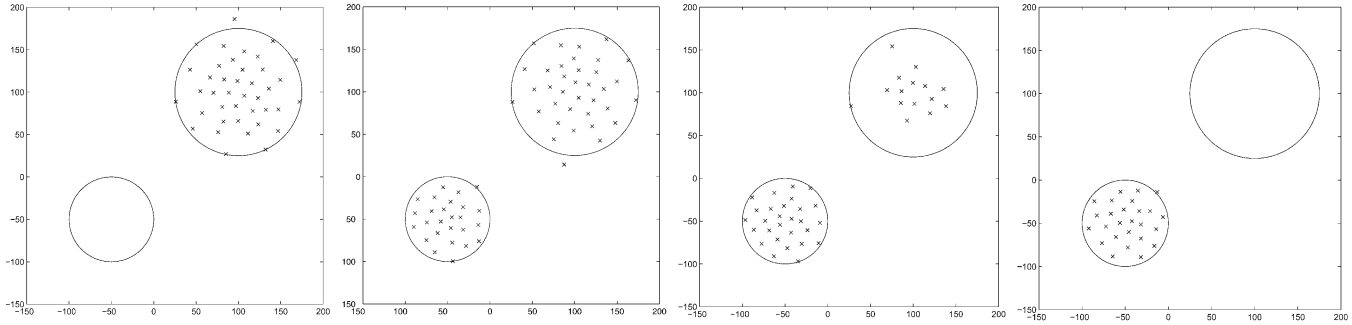


Fig. 10. Handling nonstationary input by the BCL, after (a) 20 000, (b) 40 000, (c) 80 000, and (d) 90 000 data presentations. The network size (or the cluster number) at the end of the data presentation is (a) $S_c = 40$, (b) $S_c = 64$, (c) $S_c = 43$, or (d) $S_c = 26$, respectively.

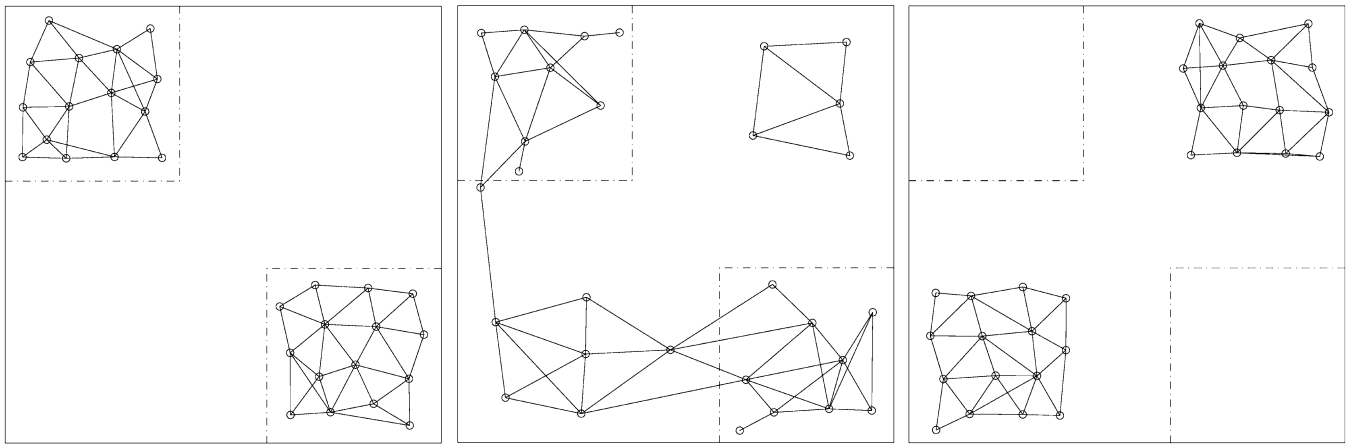


Fig. 11. Handling nonstationary input by the GNG_U after (a) 20 000, (b) 30 000, and (c) 40 000 data presentations.

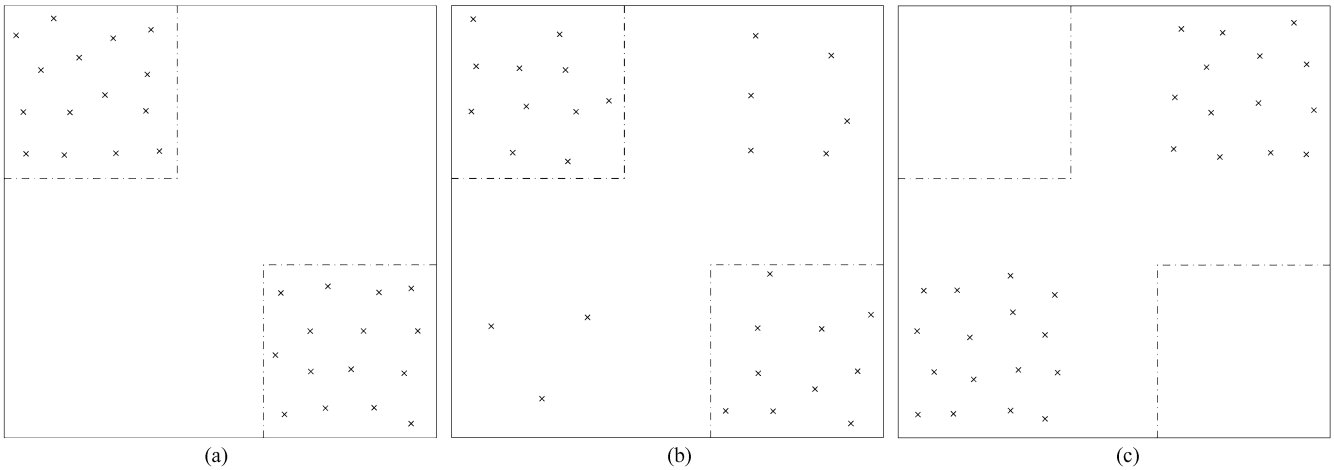


Fig. 12. Handling nonstationary input by the BCL after (a) 20 000, (b) 40 000, and (c) 60 000 data presentations.

We see that, although both the algorithms can track the nonstationary input, the GNG_U shows more adaptability to the changes in the input statistics.

V. Codebook Design Using the BCL Network

In this section, we compare the performance of the BCL algorithm with some conventional algorithms, such as the GLA, SCL, FSCL, and SOFM, on the application of codebook design for image compression. We also compare the BCL network with some self-creating models, namely, the SCONN2 [12], the modified GCS [14], [21], GNG [15], and the GNG_U [16]. Two sets of images are used as the training data sets.

1) Set-I: Lena, Boat, Baboon, Pepper, and one of the Brodatz texture images, each used separately as in [23].

Fig. 13. Training images in Set-II.

2) Set-II: A single data set consisting of eight images, as shown in Fig. 13. In this case, the test images are selected from the training image set.

All images have been taken from the USC-SIPI Image Database (http://sipi.usc.edu/services/database/) and are of size 256×256 with 256 gray levels. Each image is divided into 4096 disjoint image blocks with 4×4 pixels and these image blocks make up a 16-dimension data set. The problem of codebook design here requires that these 4096 blocks data for Set-I or 32 768 blocks data for Set-II be quantized by S_c blocks data, where S_c, the codebook size, is either pre-specified or adaptively determined according to image data distribution. As usual, we use the MSE criterion to evaluate all the codebooks, $\text{MSE} = (1/256^2)\sum_{i,j}(x(i,j) - \overline{x}(i,j))^2$, where x refers to the original image, and $\overline{x}$ to the reconstructed image.

A. Codebook Design With Fixed Book Size

1) Comparison With Some Conventional Algorithms: In the first experiment, for Set-I, the codebook size S_c is set to 64, 128, 256, 512, or 1024, respectively; while for Set-II, S_c is set to 256, 512, 1024, 2048, or 4096, repectively. To obtain relatively better results, we adopt a piecewise random scheme to initialize the codebooks of SCL, FSCL, and SOFM; that is, we first uniformly divide the data into S_c segments, then randomly select a data sample from each segment to construct the initial codebook. Experimental results show that a better codebook can be obtained by initializing the code vectors with the piecewise random scheme than with randomly selected points in the weight spaces. For GLA, the code vectors are initialized by two schemes: the first one, denoted by GLA1, is the piecewise random scheme; the other one, denoted by GLA2, is the splitting scheme as in [7]. As for the BCL, we always initialize the first code vector or the "seed" by the origin point.

In all of the algorithms based on competitive learning, the winner's learning rates decrease according to (3) with $\alpha_c(0) = 0.2$, while the neighbors' learning rate for SOFM also decreases according to (3), but with $\alpha_\gamma(0) = 0.02$. The neighbor function of SOFM is adopted as $20.0(1.0 - (t/T))$, where t represents the current competition and T denotes a pre-specified total number of data presentations. As in [7], we adopt the "fractional drop of distortion" in two consecutive iterations as the convergence indicator for the GLA algorithm, i.e., if $(D_1 - D_2)/D_2 < 0.0005$, then stop the iterations of the GLA, D_1 and D_2 being the two values of the MSE in the previous and the current iterations. For other algorithms, we simply stop the competitive learning when the total number of data presentations reaches a pre-specified value. The pre-specified number is in the form of mN, where m, called the "epoch" is a positive integer and N denotes the total number of samples in the data set. In our experiments, the epoch is always set to 20. For the BCL, the distance threshold is set as $d_0 = 20.0$, when the codebook size $S_c < 1024$ and as $d_0 = 15.0$, when $S_c \geq 1024$. The activation level threshold are set as

$$L_0 = \frac{N}{2 \times \text{Codebook Size}}.$$

The other parameters are set as $\Delta L = 1.0$, $\beta = 1/6$, and $\lambda = 1/N$. Experimental results of the average MSE and the average CPU (Pentium III 800) running time in ten consecutive trials are shown in Table I for the Lena image of Set-I. Fig. 14 shows the comparisons of the BCL with other algorithms in term of the average MSE for the Boat and Pepper images. Table II shows the experimental results for Set-II, where the test images are the Baboon and the Pepper.

From these experimental results, we can see that the BCL algorithm always provides a better codebook with a lower MSE than the other algorithms (especially for a relatively large codebook), and it, at the same time, requires a computation time no more than that of the SOFM algorithm. As a byproduct, we find that, with the first codebook initialized by the piecewise random scheme, the performance of the SCL is often better than those of the FSCL and SOFM. This seems to conflict with the general proposition that the FSCL or SOFM outperforms the SCL in competitive learning. This proposition is absolutely true, if the codebook is initialized by a set of randomly selected points from the weight space, due to the problem of *dead units*. However, if we properly initialize the codebook, e.g., the scheme in [22], the SCL often performs better than the FSCL.

2) Comparison With Other Self-Creating Models: In this section, we compare the BCL model with three other self-creating models, the SCONN2 [12], the modified GCS [21], the GNG [15], and the GNG_U [16], for the task of codebook design. According to the study in [21], the GCS has the best performance when the forgetting factor is set to zero and the so-called "η-free" node removal scheme is adopted, i.e., nodes that never win during a complete "epoch" should be deleted. Besides, some modifications in the original GCS scheme [13] are required for applying it to vector quantization.

1) Replacing the constant increment s with the input square error $\|\vec{x}(t) - \vec{\omega}_c(t)\|^2$ when updating the winner's signal counter.
2) Relaxing the structural consistency requirement to allow isolated nodes.

As in [21], we denote the modified GCS scheme by GCS_2 and set the parameters with $\epsilon_b = 0.24$, $\epsilon_n = 0.0$, $\lambda = 100$, and $\alpha = 0$, where ϵ_b and ϵ_n are respectively the winner's and neighbors' learning rates. For the SCONN2, except for the winner's learning rate, which is now set to 0.24 for fairness, the other

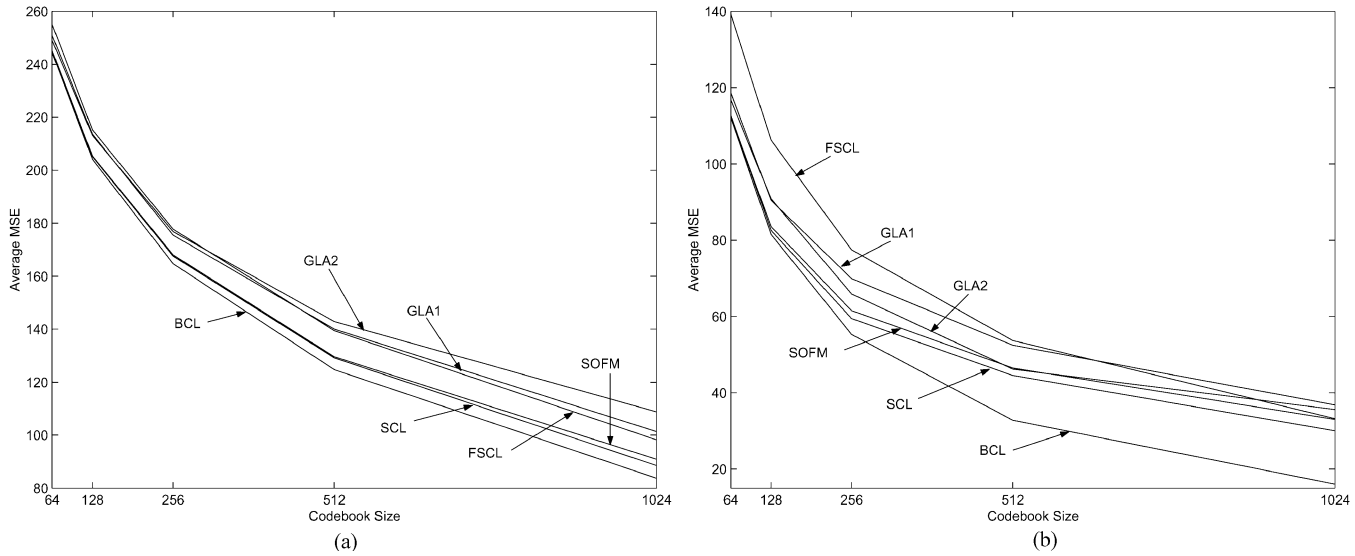


Fig. 14. Comparison of the BCL with other algorithms in term of average MSE. (a) For the Pepper image. (b) For the Baboon image.

TABLE I
PERFORMANCE COMPARISON FOR CODE BOOK DESIGN (FOR LENA IMAGE)

Codebook Size		64	128	256	512	1024
GLA1	MSE	99.29	76.28	58.68	45.21	31.01
	time(s)	6.13	10.68	18.51	33.19	63.57
GLA2	MSE	101.33	76.16	56.87	41.29	31.04
	time(s)	6.81	13.34	28.73	42.18	67.55
FSCL	MSE	115.99	89.97	66.62	46.49	28.72
	time(s)	4.12	8.20	16.31	32.57	93.83
SCL	MSE	94.53	69.83	50.98	37.82	25.90
	time(s)	4.01	7.91	15.65	31.25	87.63
SOFM	MSE	94.38	69.42	52.58	39.97	28.51
	time(s)	9.08	18.29	36.64	73.44	176.73
BCL	MSE	94.25	68.38	46.52	28.08	12.83
	time(s)	5.03	11.10	30.82	70.97	200.96

TABLE II
PERFORMANCE COMPARISON IN TERMS OF MSE FOR CODE BOOK DESIGN USING IMAGE DATA SET-II

Codebook Size		256	512	1024	2048	4096
GLA1	Baboon	217.21	190.88	160.25	131.50	103.28
	Pepper	80.60	69.65	55.29	46.45	37.57
GLA2	Baboon	225.99	192.75	163.75	137.39	104.43
	Pepper	82.75	66.93	55.32	44.07	35.16
FSCL	Baboon	230.31	201.09	172.42	140.59	107.77
	Pepper	88.30	70.95	56.36	44.98	33.83
SCL	Baboon	217.32	186.13	151.98	116.38	84.57
	Pepper	80.96	65.11	53.04	42.81	33.58
SOFM	Baboon	218.97	185.07	152.43	120.52	96.54
	Pepper	81.01	65.72	53.99	43.90	36.02
BCL	Baboon	216.46	181.95	145.22	108.64	73.54
	Pepper	78.42	64.48	52.15	39.41	28.69

parameters are set to the same values as in [12]. For the GNG and GNG_U, we set their common parameters as: $\epsilon_b = 0.24$, $\epsilon_n = 0.0006$, $\lambda = 500$, $a_{\max} = 120$, and $\beta = 0.0005$. The "age" increment for all the "edges" emanating from the winner is also set to 1.0. As for the parameter k of the GNG_U, we use a value such that it can make the network size reach the pre-specified values of the codebook size. Specifically, we set k to 30 when the codebook size is 64, and to 117 when the the codebook size is 256. For the BCL, all of the parameters take the same value as in the Section V-A.I apart from the initial learning rate, which is now set to 0.24.

Tables III and IV show the average results in ten consecutive experiments on the Lena, Baboon and Pepper images, when the codebook size is set as $S_c = 64$ and $S_c = 256$, respectively. We can see that the BCL model outperforms all the other self-creating models, including the GNG_U, in producing better code vectors with a much lower MSE.

B. Adaptive Determination of the Codebook Size

In this section we introduce an adaptive vector quantization, in which the codebook size is not pre-specified, but is adaptively determined by the image data distribution. In the experiment, the learning rate in the BCL is fixed at $\alpha_c = 0.2$ in the growing phase, during which the network keeps on growing until its size reaches a dynamical equilibrium. We employ a simple criterion to decide if the network size reaches the dynamical equilibrium: when the fluctuation of the network size S_c satisfies $|S_c(t_2) - S_c(t_1)|/S_c(t_2) \leq 0.01$ in three consecutive "epochs," we consider the network size to have reached its dynamical equilibrium, where $S_c(t_1)$ and $S_c(t_2)$ denote the two values of the codebook size at end of two consecutive "epochs." Following this, the network simply modifies the code vectors to approximate cluster centroids. In this procedure, the learning rate is adopted according to (3) with $\alpha_c(0) = 0.2$. The distance threshold is set as $d_0 = 30.0$, the activation level threshold L_0 is set at 15.0, and the other parameters are set as in Section V-A, including the GNG and GNG_U.

Fig. 15(a) shows three images with different visual complexity. Fig. 15(b)–(d) presents the growing phase of the BCL network, it is seen that the final values of the codebook size determined by the BCL is in conformity with the expectation that the higher the visual complexity of the images, the larger the size of the codebook, and Fig. 15(e) shows the reconstructed or decoded images under different codebook size.

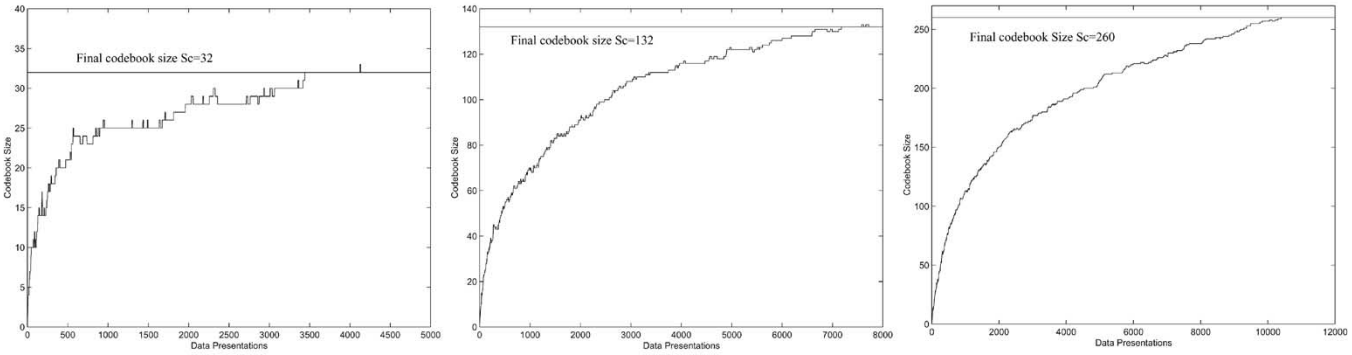
TABLE III
COMPARISON OF SELF-CREATING MODELS FOR CODEBOOK DESIGN (CODEBOOK SIZE $S_c = 64$)

	Baboon		Lena		Pepper	
	MSE	time(secs)	MSE	time (secs)	MSE	time (secs)
SCONN2	283.704	5.87	117.739	5.88	137.425	5.85
GCS_2	276.693	4.83	109.920	4.91	132.809	4.77
GNG	281.531	6.64	110.523	6.37	132.084	6.32
GNG_U	278.425	6.92	112.395	7.37	132.557	6.78
BCL	245 166	5 28	94 679	5 26	112 679	4 89

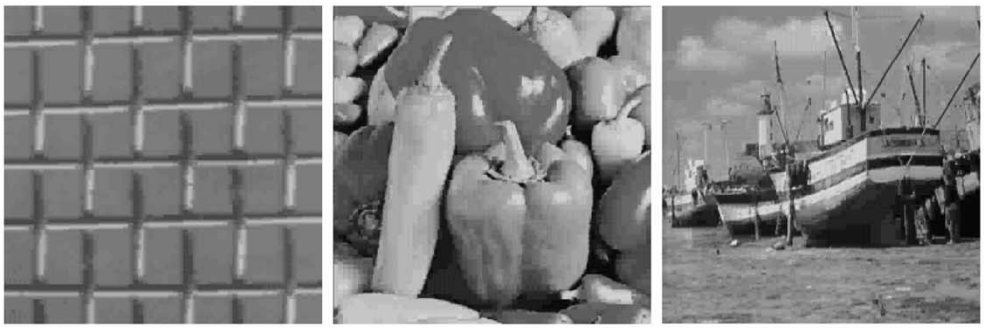
TABLE IV
COMPARISON OF SELF-CREATING MODELS FOR CODEBOOK DESIGN (CODEBOOK SIZE $S_c = 256$)

	Baboon		Lena		Pepper	
	MSE	time(secs)	MSE	time (secs)	MSE	time (secs)
SCONN2	191.367	23.40	60.327	23.43	69.011	23.39
GCS_2	188.319	21.26	54.237	36.10	64.426	37.41
GNG	187.167	31.86	53.605	27.02	63.737	26.91
GNG_U	188.715	41.42	53.839	54.72	62.137	48.34
BCL	165.187	18.73	45.928	27.69	54.262	25.70

(a) Original images.



(b) Adaptively determined codebook size.



(c) Reconstructed images.

Fig. 15. Adaptively determined codebook size and the reconstructed images.

VI. Conclusion

In this paper, we have presented a new self-creating model of competitive learning neural network by adding a special branching mechanism to the simple competitive learning model. Unlike other self-creating models, the BCL model adopts a different branching criterion, which is mainly based on two geometrical measurements of the synaptic vectors' movement in the weight space. Due to the special branching criterion, the BCL network is more efficient to capture the spatial distribution of the input data, and therefore, tends to present better clustering or quantization results. Besides, the BCL network can appropriately estimate the cluster number in a data set, adaptively respond to nonstationary data inputs, and easily lead to a multiresolution data clustering. Experimental results for vector quantization of image coding demonstrate its effectiveness and adaptability in comparison with other codebook design schemes.

References

[1] S. C. Ahalt, A. K. Arishnamurty, P. Chen, and D. E. Melton, "Competitive learning algorithms for vector quantization," *Neural Networks*, vol. 3, pp. 277–291, 1990.

[2] N. B. Karayiannis, "A methodology for constructing fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Networks*, vol. 8, pp. 505–518, May 1997.

[3] N. B. Karayiannis and P.-I. Pai, "Fuzzy algorithms for learning vector quantization," *IEEE Trans. Neural Networks*, vol. 7, pp. 1196–1211, Sept. 1996.

[4] I. Pitas, C. Kotropoulos, N. Nikolaidis, R. Yang, and M. Gabbouj, "Order statistics learning vector quantizer," *IEEE Trans. Image Processing*, vol. 5, pp. 1048–1053, June 1996.

[5] E. Yair, K. Zeger, and A. Gersho, "Competitve leaning and soft competition for vector quantization design," *IEEE Trans. Signal Processing* , vol. 40, pp. 294–309, Feb. 1992.

[6] C. Zhu and L.-M. Po, "Minimax partial distortion competitive learning for optimal codebook design," *IEEE Trans. Image Processing*, vol. 7, pp. 1400–1409, Oct. 1998.

[7] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.

[8] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[9] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, vol. COM-28, pp. 84–94, Jan. 1980.

[10] T. Kohonen, *Self-Organization and Associative Memory*. New York: Springer-Verlag, 1984, vol. 8, Springer Ser. Inform. Sci..

[11] D. Desieno, "Adding a conscience to competitive learning," in *Proc. IEEE Int. Conf. Neural Networks*, vol. I, New York, July 1988, pp. 117–124.

[12] D. I. Choi and S. H. Park, "Self-creating and organizing neural network," *IEEE Trans. Neural Networks*, vol. 5, pp. 561–575, July 1994.

[13] B. Fritzke, "Growing cell structures a self-organizing neural networks for unsupervised and supvised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[14] J.-H. Wang and W.-D. Sun, "Online learning vector quantization: A harmonic competition approach based on conservation network," *IEEE Trans. Syst., Man, Cybern.—Part B: Cybern.*, vol. 29, pp. 642–653, Oct. 1999.

[15] B. Fritzke, "A growing neural gas network learns topologies," *Adv. Neural Inform. Processing Syst.*, vol. 7, pp. 625–632, 1995.

[16] ——, "A self-organizing network that can follow nonstationary distributions," in *Proc. ICANN'97: Int. Conf. Artificial Neural Networks* , Lausanne, Switzerland, Oct. 1997, pp. 613–618.

[17] B. Kosko, "Stochastic competitive network," *IEEE Trans. Neural Networks* , vol. 2, pp. 522–529, Sept. 1991.

[18] I. King and H. Xiong, "Branching competitive learning for clustering," in *Proc. Int. Conf. Neural Information Processing (ICONIP2000)* , Taejon, Korea, Nov. 2000.

[19] H. Xiong, I. King, and Y. S. Moon, "An adptive codebook design using the branching competitve learning network," in *Proc. Int. Joint Conf. Neural Network 2001 (IJCNN'01)* , vol. 3, Washington, DC, 2001, pp. 2116–2121.

[20] B. Mirkin, *Mathematical Classification and Clustering*. Norwell, MA: Kluwer, 1996, vol. 11, Noncovex Optimizat. Applicat..

[21] J.-H. Wang, J.-D. Rau, and C.-Y. Peng, "Toward optimizing a self-creating neural network," *IEEE Trans. Syst., Man, Cybern.—Part B: Cybern.*, vol. 30, no. 4, pp. 586–593, Aug. 2000.

[22] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "A new initialization technique for generalized lloyd iteration," *IEEE Signal Processing Lett.*, vol. 1, pp. 144–146, Oct. 1994.

[23] D. Lee, S. Baek, and K. Sung, "Modified k-means algorithm for vector quantizer design," *IEEE Signal Processing Lett.*, vol. 4, pp. 2–4, Jan. 1997.

[24] N. Nasarabadi and R. A. King, "Image coding using vector quantization: A review," *IEEE Trans. Commun.*, vol. 36, pp. 957–971, Aug. 1988.

Huilin Xiong received the B.Sc and M.Sc. degrees in applied mathematics from Wuhan University, Wuhan, China, in 1985 and 1988, respectively, and the Ph.D. degree in pattern recognition and intelligent control from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, China, in 2000.

Currently, he is a Postdoctoral Researcher in the Department of Electrical and Computer Engineering, Concordia University, Quebec, Canada. His research interests include neural networks, machine learning, and wavelet-based image analysis and compression.

M. N. S. Swamy (S'59–M'62–SM'74–F'80) received the B.Sc. (Hons.) degree in mathematics from Mysore University, India, in 1954, the Diploma degree in electrical communication engineering from the Indian Institute of Science, Bangalore, India, in 1957, and the M.Sc. and Ph. D. degrees in electrical engineering from the University of Saskatchewan, Saskatoon, Canada, in 1960 and 1963, respectively.

He is presently a Research Professor and the Director of the Center for Signal Processing and Communications in the Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada, where he served as the Chair of the Department of Electrical Engineering from 1970 to 1977 and Dean of Engineering and Computer Science from 1977 to 1993. Since July 2001, he has held the Concordia Chair (Tier I) in Signal Processing. He has also taught in the Electrical Engineering Department of the Technical University of Nova Scotia, Halifax, and the University of Calgary, Calgary, as well as in the Department of Mathematics at the University of Saskatchewan. He has published extensively in the areas of number theory, circuits, systems and signal processing, and holds four patents. He is the coauthor of two book chapters and three books *Graphs, Networks and Algorithms* (New York: Wiley, 1981), *Graphs: Theory and Algorithms* (New York: Wiley, 1992), and *Switched Capacitor Filters: Theory, Analysis and Design* (Englewood Clifs, NJ: Prentice-Hall, 1995). A Russian translation of the first book was published by Mir Publishers, Moscow, Russia, in 1984, while a Chinese version was published by the Education Press, Beijing, China, in 1987.

Dr. Swamy is a Fellow of the Institute of Electrical Engineers (U.K.), the Engineering Institute of Canada, the Institution of Engineers (India), and the Institution of Electronic and Telecommunication Engineers (India). Presently, he is the President-Elect for the CAS Society. He has served the IEEE CAS Society in various capacities such as Vice President (Publications) during 2001–2002 and as Vice-President in 1976, Editor-in-Chief of the TRANSACTIONS ON CIRCUITS AND SYSTEMS-I during 1999–2001, Associate Editor of the TRANSACTIONS ON CIRCUITS AND SYSTEMS from 1985 to 1987, Program Chair for the 1973 IEEE CAS Symposium, General Chair for the 1984 IEEE CAS Symposium, Vice-Chair for the 1999 IEEE CAS Symposium and a member of the Board of Governors of the CAS Society. He is a member of Micronet, a National Network of Centers of Excellence, Canada, and also its coordinator for Concordia University. He is the recipient of many IEEE-CAS Society Awards, including the Education Award in 2000, Golden Jubilee Medal in 2000, and the 1986 Guillemin-Cauer Best Paper Award. In August 2001, he was awarded a Doctor of Science degree (*Honoris Causa*) in Engineering by Ansted University "In recognition of his exemplary contributions to the research in electrical and computer engineering and to engineering education, as well as his dedication to the promotion of signal processing and communications applications."

M. Omair Ahmad (S'69–M78–SM'83–F'01) received the B.Eng. degree from Sir George Williams University, Montreal, QC, Canada, and the Ph.D. degree from Concordia University, Montreal, QC, Canada, both in electrical engineering.

From 1978 to 1979, he was a Member of the Faculty of New York University College, Buffalo, NY. In September 1979, he joined the Faculty of Concordia University, where he was an Assistant Professor of computer science. Subsequently, he joined the Department of Electrical and Computer Engineering of the same university, where presently he is a Professor and Chair of the department. He is a researcher in the Micronet National Network of Centers of Excellence and was previously an Examiner of the Order of Engineers of Quebec. He has published extensively in the area of signal processing and holds three patents. His current research interests include the areas of multidimensional filter design, image and video signal processing, nonlinear signal processing, communication DSP, artificial neural networks, and VLSI circuits for signal processing.

Dr. Ahmad was an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: FUNDAMENTAL THEORY AND APPLICATIONS from June 1999 to December 2001. He was the Local Arrangements Chairman of the 1984 IEEE International Symposium on Circuits and Systems. During 1988, he was a member of the Admission and Advancement Committee of the IEEE. Presently, he is the Chairman of the IEEE Circuits and Systems Chapter (Montreal Section). He is a recipient of the Wighton Fellowship from the Sandford Fleming Foundation.

Irwin King received the B.Sc. degree in engineering and applied science from California Institute of Technology, Pasadena, CA, in 1984 and the M.Sc. and Ph.D degrees in computer science from the University of Southern California, Los Angeles, in 1988 and 1993, respectively.

He joined The Chinese University of Hong Kong, in 1993. He is a member of the Editorial Board of the *Neural Information Processing-Letters and Reviews Journal (NIP-LR)*. His research interests include content-based retrieval methods for multimedia databases, distributed multimedia information retrieval in peer-to-peer systems, and statistical learning theory.

He is a member of ACM, IEEE Computer Society, and the International Neural Network Society (INNS). He is also a governing board member of the Asian Pacific Neural Network Assembly (APNNA). He has served as program and/or organizing member in international conferences and workshops, e.g., WWW, ICASSP, IJCNN, ICONIP, etc. He has also served as reviewer for international conferences as well as journals, e.g., IEEE TRANSACTIONS ON NEURAL NETWORKS, IEEE PATTERN ANALYSIS AND MACHINE INTELLIGENCE, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON SYSTEM, MAN, AND CYBERNETICS, etc.