Heuristic Pattern Correction Scheme Using Adaptively Trained Generalized Regression Neural Networks

Tetsuya Hoya and Jonathon A. Chambers, *Senior Member, IEEE*

***Abstract*—In many pattern classification problems, an intelligent neural system is required which can learn the newly encountered but misclassified patterns incrementally, while keeping a good classification performance over the past patterns stored in the network. In this paper, an heuristic pattern correction scheme is proposed using adaptively trained generalized regression neural networks (GRNNs). The scheme is based upon both network growing and dual-stage shrinking mechanisms. In the network growing phase, a subset of the misclassified patterns in each incoming data set is iteratively added into the network until all the patterns in the incoming data set are classified correctly. Then, the redundancy in the growing phase is removed in the dual-stage network shrinking. Both long- and short-term memory models are considered in the network shrinking, which are motivated from biological study of the brain. The learning capability of the proposed scheme is investigated through extensive simulation studies.**

***Index Terms*—Generalized regression neural networks (GRNNs), incremental learning, pattern classification, pattern correction.**

I. Introduction

NEURAL networks have been successfully used in many pattern classification tasks [1]. Incremental training is an efficient learning mechanism for neural networks that adds new knowledge without reinitializing the entire network. The development of promising incremental learning methods has therefore been an issue with great interest in the study of neural networks [2]–[10].

In the last decade, many successful applications using the family of radial basis function neural networks (RBF-NNs) [11], [12] for the development of incremental training systems have been reported [2]–[6], [10]. In [2], incremental training is achieved by adding RBFs into the network and then adjusting their shape parameters. In contrast, in [3] a new RBF is created as a mixture of Gaussian distributions and this learning method is applied to multilinguistic handwritten character recognition. In recent work [10], a different incremental learning technique was proposed, in which new patterns are included with a relearning of the interfered patterns retrieved from past input patterns stored in the network.

Probabilistic neural networks (PNNs) [13] and generalized regression neural networks (GRNNs) [14] are the paradigms of RBF-NNs and share a special property, namely that they do not require iterative training; the weight vector between the RBFs and the output unit can be fixed as the target vector. This attractive property is particularly useful in online use of the pattern classifier, as incremental operation may be quickly achieved. Therefore, for application to online pattern correction of the misclassified patterns, the use of these networks is suitable since, in practice, the size of the incoming data set is normally very large.

In this paper, an heuristic online batch pattern correction scheme is proposed based upon a GRNN with both network growing and dual-stage shrinking mechanisms.

In the network growing phase, a subset of the misclassified patterns in the incoming data set available at cycle n is added into the network until there is no classification error within the incoming data set. Then, the grown number of centroids is reduced in the dual-stage shrinking phase. In the shrinking mechanism, a new concept of both long and short-term centroids is introduced. Moreover, the short-term centroid sets form a layered shape, representing a more hierarchical memory structure.

The proposed scheme, unlike the Parzen classifier-based approaches in [18], [19], takes an *instance-based* approach with the aid of an hierarchical data partitioning mechanism, which eliminates the need for statistical density approximation and its associated considerable mathematical complexity.

In the next section, the heuristic pattern correction scheme using a GRNN is described. The network growing and shrinking mechanisms are described in detail in Sections III and IV. Section V is devoted to the simulation studies of the proposed scheme using three different data sets for pattern classification tasks from different domains and the learning capability of the proposed scheme is evaluated through extensive experimental results.

II. The Heuristic Pattern Correction Scheme

The proposed heuristic pattern correction scheme is based upon a dual network reconfiguration process. The first stage involves network growing, in which a subset of the misclassified patterns in the incoming data set at cycle n is selected and added into the network. In the second stage, the network is reconfigured by a dual-stage network shrinking mechanism.

Manuscript received August 10, 1999; revised March 30, 2000 and July 10, 2000.

T. Hoya is with the Laboratory for Advanced Brain Signal Processing, BSI Riken, Wakoh-City, Saitama 351-0198 Japan (e-mail: hoya@bsp.brain.riken.go.jp).

J. A. Chambers was with the Communications and Signal Processing Group, Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, University of London, SW7 2BT U.K. He is now with the Department of Electronic and Electrical Engineering, University of Bath, Bath BA2 7AY, U.K. (e-mail: j.a.chamgers@bath.ac.uk).

Publisher Item Identifier S 1045-9227(01)00753-6.

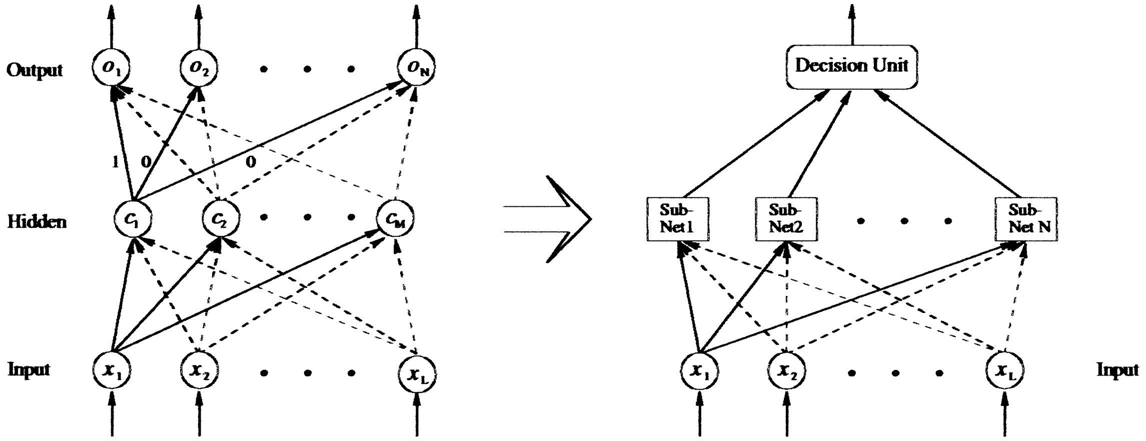

Fig. 1. Illustration of topological equivalence between the ML-GRNN with m Hidden and n output units and the assembly of the n distinct subnets.

According to biological study [21], memory in the brain can be divided into two different types, i.e., long- and short-term memory, depending on the retention time. In [21] and [11], it is also highlighted that long-term memory represents knowledge stored in the brain for a long time or permanently, while short-term memory is a compilation of knowledge representing the "current" state of the environment.

In the proposed pattern correction scheme, the concept of both long- and short-term memory models are, therefore, considered and realized in terms of leakage in the information in the network represented by the centroids.

The skeleton of the scheme is described as follows.

Skeleton of the Pattern Correction Scheme:

Step 1) *Initial Setup:* Configure the network with a set of N_l long-term centroids $C = \{c_1, c_2, ...c_{N_l}\}$ from the training set. Set cycle $n = 1$ and the short-term centroid set count $i = 1$.

Step 2) *Network Growing:* Select a subset of the misclassified patterns by testing the incoming pattern data set available at cycle n. Then, perform network growing until there is no classification error. This provides a set G_i of N_{G_i} short-term centroids from the misclassified patterns.

Step 3) *Network Shrinking:*

1) *Long-Term Memory Update:* If the cycle is a multiple of p (i.e., n mod $p = 0$), or when the total number of the centroids in the network

$$N_{total} = N_l + N_g, \qquad \text{where } N_g = \sum_{j=1}^{p} N_{G_j} \tag{1}$$

reaches/exceeds a given threshold $N_{total,\max}$, shrink the total number of centroids in the network by employing a data-pruning method and obtain N_l new long-term centroids. Reset the set count $i = 1$.

2) *Short-Term Memory Leakage (Memory Forgetting):* Otherwise, shrink only the sizes of the short-term centroid sets G_j $(j = 1, 2, \ldots, i)$. Set $i \leftarrow i + 1$.

Step 4) Set $n \leftarrow n + 1$, then return to Step 2).

In Step 2 above, the short-term centroid sets G_i $(i = 1, 2, \ldots, p)$ thus form a layered shape, representing a hierarchical memory structure. This partitioning basis has an advantage for giving a clear representation of the data stored at each correction cycle and, at the network shrinking phase, the removal of the redundancy (i.e., the least contributing centroids described later) in the short-term memory can be efficiently done.

III. The Network Growing Mechanism

The network growing in the proposed scheme consists of expanding the current network such that the grown network can correctly classify all the patterns in the currently available incoming data set.

To do this online,[1] we exploit the special property of GRNNs, namely, that, for a newly added RBF, the weight vector between the hidden layer and the output can be fixed to the target vector.

A. Network Setting for Pattern Correction

For the proposed pattern correction scheme, a fully connected multilayered GRNN (ML-GRNN) is used, which has L input neurons, M RBFs, and N output neurons. As illustrated in the left part of Fig. 1, the structure of the ML-GRNN is similar to a well-known multilayered perceptron neural network [11] except RBFs are used in the hidden layer and linear functions in the output layer. In the figure, x_i $(i = 1, 2\ldots, L)$ denote the elements in the input vector $\boldsymbol{x}$, c_j $(j = 1, 2, \ldots, M)$ (where the number of the RBFs M varies during the network reconfiguration) for the RBF are given by

$$c_j = \frac{1}{2\sigma^2} e^{-(\|x-w_j\|^2/2\sigma^2)} \tag{2}$$

where $\boldsymbol{w}_j$ is the centroid vector, σ is the radius, $\|\ldots\|^2$ is the squared Euclidean norm, and o_k $(k = 1, 2, \ldots, N)$ denote the output corresponding to Class k.

[1] Strictly speaking, the use of the term online in the proposed scheme may not be appropriate since the incremental operation will not be done on a pattern by pattern basis. Here, the authors just intend to emphasize an online use of the proposed scheme.

The target vector for Pattern i is given as a vector of indicator functions

$$\boldsymbol{T_i} = (\delta_1, \delta_2, \ldots, \delta_N)$$
$$\delta_j = \begin{cases} 1 & \text{if pattern } i \text{ belongs to the class (digit)} \\ & j-1 (j = 1, 2, \ldots N) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

With the setting above, the topology of the ML-GRNN with N output units can be seen as a set of N subnets with a decision unit as illustrated in the right part of Fig. 1, since the weight having the value zero can be removed from the network. Then, each subnet is viewed as a collection of RBFs which represents the entire pattern space for a single class. With the network on the right, the final decision is therefore made following the "winner-takes-all" strategy.

B. Selection of the Misclassified Patterns

To perform the incremental operation online, the selection of the misclassified patterns to be added must be done quickly. In this paper, the selection is such that the misclassified pattern which yields a minimum activation at the output neuron corresponding to the correct class number. This selection is reasonable since that pattern (or the newly added centroid vector) will reinforce the "rather weak" area-covering of the distribution. However, it is necessary to consider the case in which the newly added pattern may just be a noisy instance. In this paper, such an instance would be deleted in the dual-stage shrinking stage.

In commonly encountered pattern classification problems, the number of classes N_c is normally known *a priori*. For instance, $N_c = 10$ for the pattern data sets of the digit voice/character recognition tasks, corresponding to the digits from /ZERO/ to /NINE/. This knowledge is particularly important to grow the network so that the overall classification performance for each class should be improved *evenly*. Therefore, the maximum number of RBFs added in one correction count must be fixed to the number of classes.

In the following, a summary of the operation to select the misclassified patterns is given.

1) Selection of the Misclassified Patterns:

```
Step 1) Set $i = 0$.
Step 2) For $j = 1$ to $N_c$, do the following.
          If there is no misclassification for Class $j$, skip.
        Otherwise, select the misclassified pattern with a
        minimum activation at the output neuron for Class $j$
        among all the patterns in Class $j$, then set $i \leftarrow i+1$.
```

Finally, the pattern correction is performed as the network growing given below.

2) Network Growing Mechanism:

```
Step 1) Set the iteration count for the correction, $cnt = 1$.
Step 2) Test the performance of the GRNN with the current
        state using all the patterns in the incoming data set
        available at cycle $n$.
Step 3) Collect all the misclassified patterns in the incoming
        data set. Then choose a subset of the misclassified
        patterns according to the selection operation given
        above, and add a total of $i$ selected patterns into
        the GRNN. For each newly added RBF, the weight
        vector between the new RBF and the output neurons
        is fixed identical to the target vector of the
        corresponding misclassified pattern.
Step 4) Recalculate and fix the radii values of the centroids
        according to (4).
Step 5) Test again the performance of the refined GRNN
        with all the patterns in the incoming data set.
Step 6) If there is no misclassification, terminate. Otherwise
        $cnt \leftarrow cnt + 1$, and return to Step 2).
```

In Step 4) above, the radii values of the RBFs should also be updated in order to avoid the overlapping areas covered by the centroids. The way in which the radii values are readjusted is described next.

C. Radii Setting of the GRNN Classifier

The setting of radii values is a significant factor for the design of RBF-NNs and such determination is still an open issue [1], [11]. In the preliminary simulation studies, we also have investigated the individual setting of radii values using one-nearest neighbor [22], however, the performance using this technique did not yield better results than the radii setting with fixed values [20]. In this paper, fixed radii values for the respective RBFs are therefore used and set identical according to the following modified radii setting found in [11]:

$$\sigma = \frac{d}{N\sqrt{2M}} \tag{4}$$

where

d maximum Euclidean distance between the centroid vectors;

M number of RBF's;

N number of units in the output layer of the ML-GRNN.

In this paper, the radii values are updated during both the network growing and shrinking phase according to [4].

IV. The Network Shrinking Mechanism

In the network shrinking mechanism, the number of centroids in the network is reduced. As mentioned earlier, this mechanism models a function of memory learning in the actual brain; newly arrived information in the brain is processed through two different types of memory, i.e., long- and short-term memory. In the context of neural networks, this process is considered to "compress" the data stored in the network or, in other words, remove redundancy in the nodes.

By exploiting this concept, the following assumptions are made in this paper:

Assumption 1: The leakage in the short-term memory is more than that in the long-term memory.

Assumption 2: The long-term memory is updated periodically (as in the skeleton of the online learning scheme described previously, the period is determined by the value p).

A. Leakage in Short-Term Memory

For the leakage of the short-term memory, λ_s (s denotes "short term") least contributing centroids are removed from each short-term centroid set G_i ($i = 1, 2, \ldots, n \bmod p$) after the network growing. The removal is based upon the measurement quantified by the contribution of the centroid (CC).

For an RBF c_i

$$CC_{c_i} = \frac{1}{2\sigma^2} \sum_{j=1}^{N_{in}^k} e^{-(\|x_j^k - w_{c_i}\|^2 / 2\sigma^2)} \tag{5}$$

where

$\boldsymbol{x}_j^k$ ($j = 1, 2, \ldots, N_{\text{in}}^k$)	pattern vectors in the incoming data set which belong to Class k;
$\boldsymbol{w}_{c_i}$	centroid vector of the RBF, c_i;
σ	radius of the RBF.

Note that, for each set, the λ_s least contributing centroids are searched across all the classes.

The leakage in the short-term memory can then be summarized in the following.

Short-Term Memory Leakage (Memory Forgetting): From each short-term centroid set G_i ($i = 1, 2, \ldots, n \bmod p$), remove λ_s least contributing centroids. The number of the total centroids N_{G_i} after the removal is defined as

$$N_{G_i} = \max\{N_{G_i} - \lambda_s, 0\} \tag{6}$$

where zero gives a floor (i.e., no removable centroids).

B. Long-Term Memory Update

In contrast to the leakage in the short-term memory, all the centroids in the network are updated for the long-term memory. This update will occur either after a specific period (i.e., at a cycle where n is a multiple of p) or the total number of the centroids N_{total} reaches/exceeds the maximum number $N_{\text{total, max}}$.[2] For the update, a data-pruning method is used.

The data-pruning method (used in Step 3 of [Skeleton of the Pattern Correction Scheme]) must be selected so that the long-term centroids retain the "core" information gained during the last incoming cycles.

In other words, the role of the long-term centroids is to give a reasonably good generalization capacity as well as classification performance over the past patterns stored in the network. In contrast, the short-term centroids remove *instantly* the current least contributing centroids. By exploiting these two different types of memory, the network can be always kept in a compact size.

Moreover, with the introduction of the two-stage shrinking mechanism, the effect upon the pattern correction system of a noisy instance would also be small since, even if such an instance may temporarily be added in the network growing phase, such an instance will be removed either at the next cycle or later at the long-term memory update.

V. Simulation Study

In the simulation study, the proposed online pattern correction scheme was applied to the three different data sets, namely the SFS [23] for digit voice recognition and the two data sets, namely the OptDigit and PenDigit data set, for character recognition tasks chosen from "UCI Machine Learning Repository" of the University of California.

For the two data sets, i.e., the SFS and Pendigit, the volume was partitioned into eight distinct sets: training and testing (never used for training) and the remaining six for the incoming data set no. 1–6, while a total of 14 (one for training and testing, and the remaining 12 for the incoming) partitioned data sets were used for the OptDigit data set.

The original UCI data sets come with two distinct data sets ready for training and testing. For each UCI data set (i.e., the OptDigit and PenDigit data set), a total of 3600 feature vectors for training and incoming were arbitrarily chosen from the original training set.[3] Similarly, for testing, a total of 400 feature vectors were selected among the vectors in the original testing data set. Table I shows a list of the data sets used for the simulation study in this paper.

Moreover, in order to confirm the consistency of the simulation results, three different combinations of the (training/six incoming) data sets were tried for all the three data sets.

A. Parameter Setting for the Network Shrinking Mechanism

In the simulation, the proposed online pattern correction scheme was performed for the six (or, twelve for the OptDigit) distinct incoming sets described in the previous section [i.e., the simulation was stopped at $n = 6(12)$] and the following parameters were used.

- Maximum number of the total centroids $N_{\text{total, max}} = N_l + N_{g,\max}$, where $N_{g,\max}$ is the maximum number of the grown (short-term) centroids.
- Number of removable centroids from the short-term memory: $\lambda_s = 2$.
- Period for updating the long-term memory: $p = 2$. (For example, the update occurred three times during the simulation in this paper.[4])

In the above, the maximum number of total centroids in the network is known *a priori* and may be fixed, dependent on the application. Since this number represents the memory capacity (e.g., in practice this number is used to avoid memory overflow problem in real implementation) and gives a threshold for the additional centroids in the network growing phase. However, the choice must be dependent on the number of long-term centroids considering the generalization capability. For both the SFS and the Pendigit data sets, $N_{g,\max}$ was fixed to 100, while $N_{g,\max} = 300$ for the OptDigit data set.

Similarly, λ_s can be fixed using the *a priori* knowledge; as the value λ_s is increased, the more the network forgets the recent data. (In our examples, it was empirically found that the selection $\lambda_s = 2$ gives a reasonable tradeoff.)

In the simulation, the total number of the centroids in the network was pruned identical to the number of the initial centroids,

[2] This number corresponds to the "saturation" of memory capacity.

[3] The original OptDigit data set contains a total of 3823 and 1797 vectors for training and testing, respectively, while the original PenDigit data set consists of 7494 and 3498 vectors for training and testing, respectively.

[4] In the simulation, the long-term update period was arbitrarily chosen and fixed to $p = 2$ for all the three domain data sets. This was done in order to perform the performance comparison of the data-pruning algorithms with a smaller number of the parameters. However, different choices of p will be discussed later in this section.

TABLE I
DATA SETS USED IN THE SIMULATION STUDY

Data Set	Total Num. of Samples in the Data Set	Total Num. of Samples in the Training Set	Num. of Long Term Centroids	Num. of In-coming Data Sets
SFS	900	270	80	6
OptDigit	4000	1200	160	12
PenDigit	4000	1200	80	6

Data Set	Num. of Samples in Each In-coming Set	Num. of Samples in the Testing Set	Num. of Data Points in a Feature Vector
SFS	90	90	256
OptDigit	200	400	64
PenDigit	400	400	16

i.e., $N_{l,\,\text{new}} = N_l$ (however, the actual centroid vectors will be different from the initial setting).

The number $N_{l,\,\text{new}}$ can be varied to represent the more dynamic nature of the memory learning process and to obtain (hopefully) an improved classification and better representation of the pattern space. In concept, as in real brain tissue, modeling multistage (or nested) shrinking mechanisms can be possible. In reality, however, such dynamic configuration is very hard to analyze and is therefore not considered in this paper.

For the long-term memory update, four different data-pruning algorithms, i.e., the k-means [25], Vertex–Chain [26], List-Splitting [26], and the shortest spanning tree (SST)-Splitting algorithm [26], were used and a performance comparison is made later in this paper.

The three graph theoretic oriented algorithms in [26] are all based upon a combination of an hierarchical graph partitioning of the original graph, which is formed from all the patterns in the data set, into its subgraphs and the search for the locations of the centers [27] on each subgraph. In [28], the superiority of the three data-pruning algorithms to the k-means clustering algorithm, in terms of their both computational and classification performance over the data sets collected from two speech databases, is reported.

The algorithms differ from each other in their ways of partitioning of the original graph into its disjoint subgraphs; in Vertex–Chain algorithm, all the vertices in the original graph are first arranged on a chain, according to the distances from the ith dominant vertex ($i = 1, 2, \ldots q$, q is the counting number of partitioning.), then the chain is cut into two pieces. This process is repeated for q times to obtain a total of $2q$ disjoint subgraphs (i.e., tournament, in shape). For each subgraph, the location of the absolute center is calculated and converted into the corresponding representative pattern of the data set. The $2q$ representative patterns so obtained are therefore used for the long-term centroids in this paper.

In contrast, the original graph is *recursively* partitioned in both List-Splitting and SST-Splitting algorithms. In List-Splitting algorithm, the distance between each vertex and the most (first) dominant vertex is tabulated into an distance-order list. Then the list is split into $2^{q}-$ parts. As in VertexChain algorithm, a total of 2^{q} representative patterns are obtained from the absolute centers of the respective subgraphs. In SST-Splitting algorithm, an SST of the original graph is created as the initial partitioning target instead of the order list. After the recursive splitting, a total of q disjoint subgraphs are obtained. (Note that, unlike Vertex–Chain or List-Splitting algorithm, exactly q representative patterns can be obtained after q-times splitting, i.e., this method does not have any limit on the number of generating representative patterns.)

B. The SFS Data Set

The SFS data set consists of a total of 900 utterances of the digits from /ZERO/ to /NINE/ recorded in English by nine different speakers (including even numbers of female and male speakers). Each utterance is sampled at 20 kHz and is converted into a feature vector with a normalized set of 256 data points obtained by the well-known LPC-mel-cepstral analysis (e.g., see [] or [24]). The feature vector is therefore used as the input vector of the GRNN.

For the simulation using the SFS data set, two different configurations of the data set were considered. The first corresponds to the data set where both the training and the incoming data sets evenly contain the utterances recorded by the nine speakers (SFS Data Set 1), in the second the training set, in contrast, contains those recorded by only three speakers and each incoming set contains an unknown speaker, for modeling a more general situation (SFS Data Set 2). For both cases, the number of patterns for each digit was evenly fixed so as to make the network grow in a "well-balanced" shape.

1) Initial Choice of RBFs: The initial choice of the centroids from the training set was performed by the k-means clustering algorithm.

In the proposed shrinking mechanism, it is important to consider the ratio between the total number of long- and short-term centroids in the network. Since, as described earlier, long-term centroids contribute to the fundamental generalization capability of the network.

To confirm this, a comparison of the effect of varying the number of long-term centroids upon the pattern correction system was made, using the SFS Data Set 1. Fig. 2 shows the variation in the classification performance[5] with the number of long-term centroids chosen by the k-means clustering method fixed at 20, 40, and 80 ($n = 0$: with the initial setup, the performance is averaged over three different trials). In the figure, the classification performance varied greatly with smaller numbers of the centroids, whereas, with 80 long-term centroids, the performance becomes much more stable. In the same figure, it is interestingly observed that the performance with 80 long-term centroids is slightly improved at each long-term update.

In Fig. 3, on the other hand, the ratio between the total number of centroids in the network and long-term centroids is given. The ratio r_{lt} is simply defined as

$$r_{lt} = \frac{N_{\text{total}}}{N_l}$$

[5] In this paper, the term "classification performance" is defined as the correct classification rate over the testing set, unless explicitly denoted otherwise.

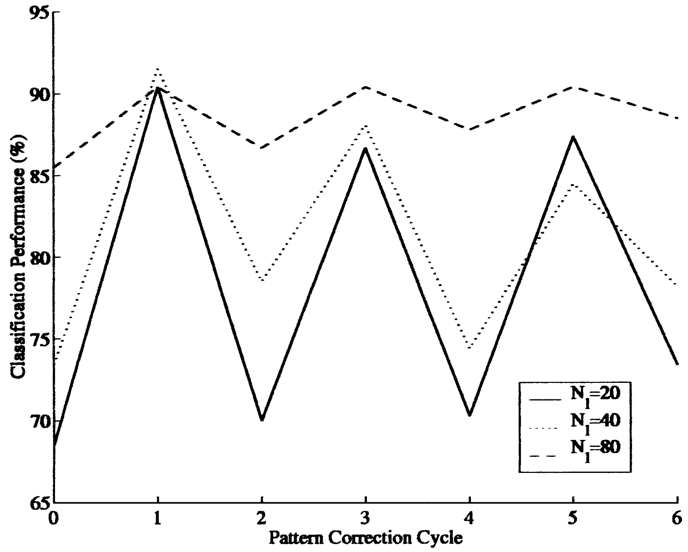

Fig. 2. Variations in terms of the classification performance over the testing set *after* the network shrinking (long-term memory update occurs at pattern correction cycles 2, 4, and 6. In the x-axis of the figure, "o" corresponds to the initial state).

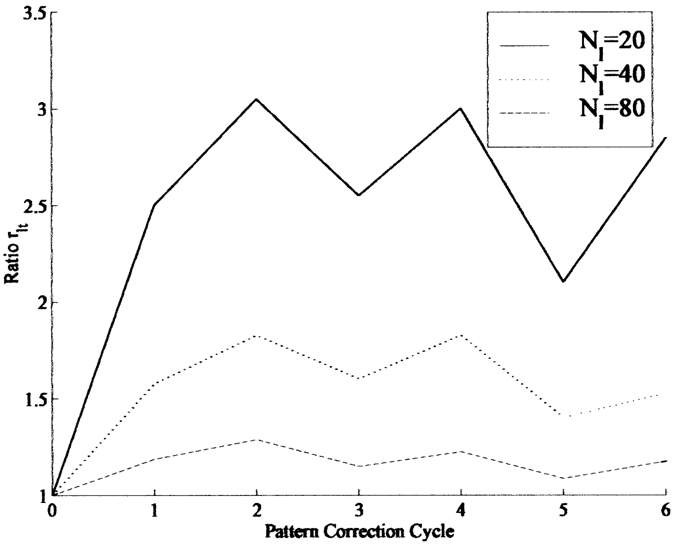

Fig. 3. Variation in terms of the ratio between the number of the long-term centroids and the total number of the centroids in the network.

$$= \frac{N_l + N_g}{N_l}. \tag{7}$$

In the figure (note that, unlike Fig. 2, the value r_{lt} is calculated before long-term memory update), the ratio r_{lt} with 40 or 80 long-term centroids becomes more steady in comparison with that of 20 centroids. This suggests that at smaller number the long-term memory is easily collapsed by the grown centroids at each pattern correction cycle, whereas, at larger number of long-term centroids, the long-term memory is not affected and the grown centroids are, in turn, considered to reinforce the classification performance.

In the simulation using SFS Data Set 1, a total of 80 long-term centroids was thus considered to be suitable for the evaluation, in terms of the generalization capability. Based upon the same principle as for the SFS Data Set 1, the number for SFS Data Set 2 was also fixed to 80.

2) Simulation Results: In Table II and IV, the variations in the total number of centroids in the network in order that a perfect pattern correction is achieved by the proposed growing mechanism are shown (the results shown are averaged over the three different trials) using SFS Data Set 1 and 2, respectively.

As shown, for the SFS Data Set 1, the numbers of centroids spread between 87 and 115 for each data-pruning algorithm, while, similar to the case using SFS Data Set 1, the numbers using the SFS Data Set 2 spread between 89 and 110. Note that, for both cases, the numbers of centroids generated by the Vertex–Chain method are always greater than the other three methods.

Table III and V, in contrast, show the averaged classification performance with the testing (unknown) data set after each shrinking phase, using SFS Data Set 1 and 2, respectively. In the tables, note that the classification performance after the long-term memory update (i.e., at $n = 2, 4$, and 6) is not degraded significantly for the case using k-means, List-Splitting, and SST-Splitting method. This indicates that the update preserves the generalization capability of the network achieved during the iterative correction cycles.

As in the tables, it is also observed that, for both cases, the overall classification performance using the three data-pruning methods, i.e., k-means, List-Splitting, and SST-Splitting, is improved from the initial setting of the network, though the performance using the Vertex–Chain method is degraded as the pattern correction cycle increases.

C. The Two UCI Data Sets

In the simulation using the OptDigit data set, a total of 160 initial long-term centroids were obtained by the k-means clustering algorithm, as for the simulation using the SFS data sets. Similarly, a total of 80 pruned vectors were used as initial long-term centroids for the PenDigit data set. The numbers of the initial long-term centroids were fixed by means of the *a priori* knowledge with the same principle as for the SFS Data Set 2 described in Section V-B1.

1) Simulation Results: Tables VI and VIII, respectively, show the variation in the total number of the centroids at the achievement of perfect pattern correction using the OptDigit and PenDigit data sets. For the OptDigit, the total number of centroids spreads between 166 and 197, while the numbers spread between 88 and 118 for the PenDigit, as for the cases using SFS Data Set no. 1 and 2.

Note that the total numbers of centroids using Vertex–Chain method are, again, always greater than those using the other three data-pruning methods.

In Table VII and IX, the classification performance with the testing data set after the shrinking phase is given using the OptDigit and PenDigit data set, respectively. In the tables, the performance using the Vertex–Chain method is degraded with increasing the pattern correction cycle as observed in the simulation using the SFS Data Set 1 and 2, while the performance using the other data-pruning methods shows an improvement over the initial network setting.

TABLE II
VARIATION IN THE TOTAL NUMBER OF THE CENTROIDS IN THE NETWORK WHEN PERFECT CORRECTION IS ACHIEVED USING SFS DATA SET (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for	Total Number of Centroids After							Average from
Long Term	the Pattern Correction at Cycle n							$n = 3$ to
Memory Update	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 6$
k-means	80	95	105	92	98	87	94	93
List-Splitting				94	98	89	96	94
SST-Splitting				92	99	87	93	93
Vertex-Chain				97	105	102	115	105

TABLE III
CLASSIFICATION PERFORMANCE OVER THE TESTING DATA SET AFTER THE SHRINKING PHASE, USING SFS DATA SET 1 (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for	Classification Performance							Average from
Long Term	After the Network Shrinking at Cycle n							$n = 1$ to
Memory Update	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 6$
k-means	85.5%	90.4%	86.7%	90.4%	87.8%	90.4%	88.5%	89.0%
List-Splitting			83.7%	88.5%	87.7%	90.4%	87.8%	88.1%
SST-Splitting			85.9%	88.2%	88.5%	90.0%	88.9%	88.7%
Vertex-Chain			80.0%	84.4%	78.9%	87.8%	77.8%	83.2%

TABLE IV
VARIATION IN THE TOTAL NUMBER OF THE CENTROIDS IN THE NETWORK WHEN PERFECT CORRECTION IS ACHIEVED USING SFS DATA SET 2 (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for	Total Number of Centroids After							Average from
Long Term	the Pattern Correction at Cycle n							$n = 3$ to
Memory Update	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 6$
k-means	80	96	108	90	103	91	105	97
List-Splitting				90	102	92	102	97
SST-Splitting				91	102	89	99	95
Vertex-Chain				95	110	96	109	103

TABLE V
CLASSIFICATION PERFORMANCE OVER THE TESTING DATA SET AFTER THE SHRINKING PHASE, USING SFS DATA SET 2 (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for	Classification Performance							Average from
Long Term	After the Network Shrinking at Cycle n							$n = 1$ to
Memory Update	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	$n = 6$
k-means	84.1%	83.7%	82.2%	87.8%	84.8%	85.6%	85.9%	85.0%
List-Splitting			87.4%	89.6%	84.4%	87.8%	87.8%	86.8%
SST-Splitting			87.8%	87.0%	89.3%	88.2%	89.6%	87.6%
Vertex-Chain			78.2%	80.8%	76.3%	79.3%	73.3%	78.6%

D. Discussion on the Results

In the simulation studies of the three different domain data sets, it has consistently been observed that the performance with three out of the four data-pruning methods (i.e., k-means, List-Splitting, and SST-Splitting method) used for updating long-term memory is improved over that of the initial setup, though the performance with the Vertex–Chain method is degraded as the pattern correction cycles increase. It has also been observed that the number of grown centroids using the Vertex–Chain method is always greater than that using the other three data-pruning methods.

These indicate that both the List-Splitting and SST-Splitting methods have the capability of refining the shape of the

TABLE VI
TRANSITION IN THE TOTAL NUMBER OF THE CENTROIDS IN THE NETWORK WHEN PERFECT CORRECTION IS ACHIEVED USING OPTDIGIT DATA SET (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Total Number of Centroids After the Pattern Correction at Cycle n						
	Init.	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
k-means	160	184	192	169	171	178	168
List-Splitting				168	175	171	173
SST-Splitting				170	175	171	176
Vertex-Chain				173	185	181	195

Method for Long Term Memory Update	Total Number of Centroids After the Pattern Correction at Cycle n						Average from $n=3$ to $n=12$
	$n=7$	$n=8$	$n=9$	$n=10$	$n=11$	$n=12$	
k-means	168	174	168	171	166	176	172
List-Splitting	171	173	167	172	167	176	171
SST-Splitting	167	171	169	172	167	175	171
Vertex-Chain	184	197	180	191	181	197	186

TABLE VII
CLASSIFICATION PERFORMANCE OVER THE TESTING DATA SET AFTER THE SHRINKING PHASE, USING OPTDIGIT DATA SET (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Classification Performance After the Network Shrinking at Cycle n						
	Init.	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$
k-means	86.8%	90.8%	90.1%	90.3%	90.5%	91.2%	90.9%
List-Splitting			89.3%	93.0%	90.7%	91.4%	91.3%
SST-Splitting			90.4%	90.6%	89.7%	90.9%	90.8%
Vertex-Chain			84.3%	86.1%	83.0%	84.1%	83.1%

Method for Long Term Memory Update	Classification Performance After the Network Shrinking at Cycle n						Average from $n=1$ to $n=12$
	$n=7$	$n=8$	$n=9$	$n=10$	$n=11$	$n=12$	
k-means	91.2%	89.1%	89.4%	89.8%	89.8%	90.4%	90.3%
List-Splitting	91.2%	89.8%	89.9%	88.9%	89.5%	88.4%	90.4%
SST-Splitting	90.7%	90.1%	90.4%	91.1%	91.2%	90.2%	90.6%
Vertex-Chain	83.7%	77.4%	79.5%	76.2%	79.1%	75.2%	81.9%

TABLE VIII
TRANSITION IN THE TOTAL NUMBER OF THE CENTROIDS IN THE NETWORK WHEN PERFECT CORRECTION IS ACHIEVED USING PENDIGIT DATA SET (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Total Number of Centroids After the Pattern Correction at Cycle n							Average from $n=3$ to $n=6$
	Init.	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	
k-means	80	103	112	91	94	88	96	93
List-Splitting				95	103	94	101	98
SST-Splitting				91	97	89	98	94
Vertex-Chain				103	108	109	118	110

pattern space spanned by the long-term centroids as well as the k-means clustering method and that the Vertex–Chain method is, however, suffering from sparse distribution of the data points which affects the overall performance [26] and

TABLE IX
CLASSIFICATION PERFORMANCE OVER THE TESTING DATA SET AFTER THE SHRINKING PHASE, USING PENDIGIT DATA SET (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Classification Performance After the Network Shrinking at Cycle n							Average from $n = 1$ to $n = 6$
	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	
k-means	88.0%	94.3%	92.8%	93.9%	90.3%	92.3%	90.3%	92.3%
List-Splitting			86.2%	91.7%	89.3%	92.6%	89.3%	90.6%
SST-Splitting			92.4%	92.5%	93.1%	93.7%	91.6%	92.9%
Vertex-Chain			82.7%	93.4%	82.1%	91.0%	74.4%	86.3%

is hence considered to be inappropriate for the shrinking mechanism.

In the simulation, the long-term update period was always fixed to $p = 2$ for all the three domain data sets since the main focus of the simulation study is to investigate the performance of the different data-pruning methods. In Table XI another performance comparison using OptDigit data set where $p = 4$ is given. In comparison of Table VII with Table XI, it is observed that the classification performance with $p = 4$ is comparable or sometimes slightly better than that with $p = 2$, at the expense of the grown number of the centroids as observed by comparing Table VI with Table X. From these observations, it can be said that the effect upon the generalization performance by means of the change in p would be relatively small, though there still may be a tradeoff between the total number of centroids and the generalization performance. Therefore, in more practical situations, the value can be fixed according to the size (if known) or the *a priori* number of the available incoming data sets.

VI. CONCLUSION

In this paper, an heuristic online pattern correction scheme using GRNNs has been proposed and applied to three data sets from different domains, i.e., the SFS and the two UCI data sets, with a variant of their initial settings. Within the proposed online batch pattern correction scheme, both the network growing and the two-stage network shrinking mechanisms have been developed.

In the simulation study, it has been shown that the misclassified patterns can be perfectly corrected by the network growing mechanism with comparably small number of centroids. This property is considered to be particularly suitable for application in strict security service systems where quick pattern correction and recognition performance without failure over a specific pattern set is desired.

In contrast, in the network shrinking phase, both long-term memory update and short-term memory leakage mechanisms have been considered based upon biological studies [21], [11] and realized in terms of the number of the centroids in the network.

For the long-term memory update, it has been found that the three data-pruning methods, i.e., k-means, List-Splitting, and SST-Splitting method, are suitable, while the Vertex–Chain method is not due to the sparse distribution problem.

TABLE X
TRANSITION IN THE TOTAL NUMBER OF THE CENTROIDS IN THE NETWORK WHEN PERFECT CORRECTION IS ACHIEVED USING OPTDIGIT DATA SET WITH $p = 4$ (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Total Number of Centroids After the Pattern Correction at Cycle n						
	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
k-means	160	184	192	198	199	168	174
List-Splitting						170	179
SST-Splitting						169	177
Vertex-Chain						180	190

Method for Long Term Memory Update	Total Number of Centroids After the Pattern Correction at Cycle n						Average from $n = 5$ to $n = 12$
	$n = 7$	$n = 8$	$n = 9$	$n = 10$	$n = 11$	$n = 12$	
k-means	179	181	170	174	177	183	176
List-Splitting	186	188	168	172	172	178	177
SST-Splitting	182	184	170	172	177	184	177
Vertex-Chain	196	201	178	187	193	204	191

TABLE XI
CLASSIFICATION PERFORMANCE OVER THE TESTING DATA SET AFTER THE SHRINKING PHASE, USING PENDIGIT DATA SET WITH $p = 4$ (THE RESULTS ARE AVERAGED OVER THREE DIFFERENT TRIALS)

Method for Long Term Memory Update	Classification Performance After the Network Shrinking at Cycle n						
	Init.	$n = 1$	$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$
k-means	86.8%	90.8%	92.3%	91.2%	90.9%	91.6%	91.5%
List-Splitting					89.4%	89.9%	89.4%
SST-Splitting					90.9%	91.4%	91.3%
Vertex-Chain					83.1%	82.1%	85.6%

Method for Long Term Memory Update	Classification Performance After the Network Shrinking at Cycle n						Average $n = 1$ to $n = 12$
	$n = 7$	$n = 8$	$n = 9$	$n = 10$	$n = 11$	$n = 12$	
k-means	91.7%	91.2%	91.5%	91.5%	91.4%	91.2%	91.4%
List-Splitting	89.9%	90.2%	90.2%	89.9%	90.1%	90.3%	90.3%
SST-Splitting	91.6%	92.7%	92.7%	92.7%	92.5%	91.7%	91.8%
Vertex-Chain	86.5%	80.2%	81.8%	83.7%	84.2%	74.8%	84.7%

Future work will be directed toward the development of the integrated algorithms/mechanisms which provide a more dynamic online based pattern correction scheme by exploiting both the refining property of the k-means clustering and the hierarchical advantage of the graph theoretic methods.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their encouragement and insightful suggestions/comments.

REFERENCES

[1] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1996.

[2] M. R. Berthold and J. Diamond, "Constructive training of probabilistic neural networks," *Neurocomputing*, vol. 19, pp. 167–183, 1998.

[3] H.-C. Fu and Y. Y. Xu, "Multilinguistic handwritten character recognition by Bayesian decision-based neural networks," *IEEE Trans. Signal Processing*, vol. 46, pp. 2781–2789, Oct. 1998.

[4] L. Fu, H.-H. Hsu, and J. C. Principe, "Incremental backpropagation learning networks," *IEEE Trans. Neural Networks*, vol. 7, pp. 757–761, May 1996.

[5] L. Fu, "Incremental knowledge acquisition in supervised learning networks," *IEEE Trans. Syst., Man, Cybern. A*, vol. 26, pp. 801–809, Nov. 1996.

[6] N. B. Karayiannis and G. W. Mi, "Growing radial basis function neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Trans. Neural Networks*, vol. 8, pp. 1492–1506, Nov. 1997.

[7] S. Y. Kung and J. S. Taur, "Decision-based neural networks with signal/image classification applications," *IEEE Trans. Neural Networks*, vol. 6, pp. 170–181, Jan. 1995.

[8] C. P. Lim and R. F. Harrison, "An incremental adaptive network for online supervised learning and probability estimation," *Neural Networks*, vol. 10, no. 5, pp. 925–939, 1997.

[9] S. Shitotani, T. Fukuda, and T. Shibata, "A neural-network architecture for incremental learning," *Neurocomputing*, vol. 9, pp. 111–130, 1995.

[10] K. Yamauchi, N. Yamaguchi, and N. Ishii, "Incremental learning methods with retrieving of interfered patterns," *IEEE Trans. Neural Networks*, vol. 10, pp. 1351–1365, Nov. 1999.

[11] S. Haykin, *Neural networks: A Comprehensive Foundation*. New York: Macmillan, 1994.

[12] M. J. L. Orr. (1996, Apr.) *Introduction to radial basis function networks* [Online] Available www.cns.ed.ac.uk

[13] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, pp. 109–118, 1990.

[14] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 568–576, Nov 1991.

[15] D. M. Allen, "The relationship between variable selection and data augmentation and a method for prediction," *Technometrics*, vol. 16–1, pp. 125–127, 1974.

[16] B. Efron, *The Jacknife, the Bootstrap, and Other Resampling Plans*. Philadelphia, PA: Soc. Ind. Appl. Math., 1982, ch. 7.

[17] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *J. Roy. Statist. Soc.*, vol. 1, pp. 111–147, 1974.

[18] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[19] M. A. Kraaijveld and R. P. W. Duin, "Generalization capabilities of minimal kernel-based networks," in *Proc. Int. Joint Conf. Neural Networks*, Seattle, 1991, pp. 843–848.

[20] T. Hoya, "Graph Theoretic Methods for Data Partitioning," Ph.D. dissertation, Imperial College, London, U.K., Aug. 1997.

[21] M. A. Arbib, *The Metaphorical Brain*, 2nd ed. New York: Wiley, 1989.

[22] A. Saha and J. D. Keeler, "Algorithms for better representation and faster learning in radial basis function networks," in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, 1990, pp. 482–489.

[23] M. Huckvale, *Speech Filing System Vs3.0—Computer Tools For Speech Research*. London, U.K.: University College, Mar. 1996.

[24] S. Furui, "Cepstral analysis techniques for automatic speaker verification," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 254–272, Apr. 1981.

[25] J. R. Deller Jr, J. G. Proakis, and J. H. L. Hansen, *Discrete-Time Processing of Speech Signals* New York, 1993.

[26] J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Symp. Matho. Stat. Prob.*, vol. 1. Berkeley, CA, 1967, pp. 281–297.

[27] T. Hoya, "Graph theoretic techniques for pruning data and their applications," *IEEE Trans. Signal Processing*, vol. 46, pp. 2574–2579, Sept. 1998.

[28] N. Christofides, *Graph Theory: An Algorithmic Approach*. New York: Academic, 1975.

Tetsuya Hoya was born in Tokyo, Japan, on September 15, 1969. He received the B.Sc. and M.Sc. degrees both from Meiji University, Japan, in 1992 and 1994, respectively, in electrical engineering. He received the Ph.D. degree in signal processing from Imperial College of Science, Technology and Medicine, University of London, U.K., in 1998.

From April 1994 to September 1994, he was a Research Assistant at Department of Electronics and Communication, Graduate School of Meiji University, Japan. He was then a student at Department of Electrical and Electronics Engineering, Imperial College of Science, Technology and Medicine, from October 1994 to December 1997. He was a Postdoctoral Research Associate at Department of Electrical and Electronics Engineering, Imperial College, London, from September 1997 to August 2000. Since October 2000, he has been a Researcher within the Brain Science Institute, Riken, Japan. His research interests include adaptive signal processing with applications in communications, image segmentation, speech enhancement/recognition, intelligent neural networks for signal processing, and combinatorial optimization.

Jonathon A. Chambers (M'93–SM'99) was born in Peterborough, U.K., in 1960. After an electronics artificer apprenticeship in the Royal Navy, he received the first class B.Sc. (Hons.) degree in electronic engineering from the Polytechnic of Central London, U.K. He received the Ph.D. degree in adaptive signal processing in 1990 from Imperial College, London, U.K., and Cambridge University, Cambridge, U.K.

He spent three years as a Research Scientist at Schlumberger Cambridge Research, applying adaptive signal processing techniques to oilfield related applications. He returned to a lectureship in signal processing in the Department of Electrical and Electronic Engineering, Imperial College, in 1994 and was promoted to a readership in signal processing in 1998. He has authored and coauthored 140 technical publications in signal processing and its applications.

Dr. Chambers is a member of the IEE Professional Group E5 on Signal Processing, and has served as a Guest Editor for the International Journal of Adaptive Control and Signal Processing, and as an Associate Editor for IEEE TRANSACTIONS ON SIGNAL PROCESSING. He received the Robert Mitchell Medal as the top graduate in 1985 from the Polytechnic of Central London.