Journal of Intelligent Learning Systems and Applications, 2010, 2, 179-189
doi:10.4236/jilsa.2010.24021 Published Online November 2010 (http://www.scirp.org/journal/jilsa)

An Autonomous Incremental Learning Algorithm for Radial Basis Function Networks

Seiichi Ozawa[1], Toshihisa Tabuchi[1], Sho Nakasaka[1], Asim Roy[2]

[1]Graduate School of Engineering, Kobe University, Kobe, Japan; [2]Department of Information Systems, Arizona State University, Tempe, USA
Email: ozawasei@kobe-u.ac.jp, asim.roy@asu.edu

Received March 29th, 2010; revised September 14th, 2010; accepted September 17th, 2010.

ABSTRACT

In this paper, an incremental learning model called Resource Allocating Network with Long-Term Memory (*RAN-LTM*) *is extended such that the learning is conducted with some autonomy for the following functions:* 1) *data collection for initial learning,* 2) *data normalization,* 3) *addition of radial basis functions* (*RBFs*)*, and* 4) *determination of RBF centers and widths. The proposed learning algorithm called Autonomous Learning algorithm for Resource Allocating Network* (*AL-RAN*) *is divided into the two learning phases: initial learning phase and incremental learning phase. And the former is further divided into the autonomous data collection and the initial network learning. In the initial learning phase, training data are first collected until the class separability is converged or has a significant difference between normalized and unnormalized data. Then, an initial structure of AL-RAN is autonomously determined by selecting a moderate number of RBF centers from the collected data and by defining as large RBF widths as possible within a proper range. After the initial learning, the incremental learning of AL-RAN is conducted in a sequential way whenever a new training data is given. In the experiments, we evaluate AL-RAN using five benchmark data sets. From the experimental results, we confirm that the above autonomous functions work well and the efficiency in terms of network structure and learning time is improved without sacrificing the recognition accuracy as compared with the previous version of AL-RAN.*

Keywords: *Autonomous Learning, Incremental Learning, Radial Basis Function Network, Pattern Recognition*

1. Introduction

In general, when a learning model is applied to real-world problems, it does not always work well unless a human supervisor participates initial settings such as choosing proper parameters and selecting the data preprocessing (e.g., feature extraction / selection and data normalization) depending on given problems. This dependence on human supervision is one of the highest barriers to wide deployment of artificial learning systems. Therefore, removing or alleviating human intervention in artificial learning systems is a crucial challenge. In fact, there have been proposed several approaches to autonomous learning [1-5].

On the other hand, the radial basis function network (RBFN) is one of the most popular models that has been applied to many applications such as pattern recognition and time-series prediction [6]. Although RBFN has mainly been used as batch learning, the significance of its extension to incremental learning is growing from a practical point of view [7,8]. Especially, *one-pass incremental learning* [9] is an important concept for large-scale high-dimensional data. In this type of learning, the learner is required to acquire knowledge with a single presentation of training data (*i.e.*, training data are never presented repeatedly for retraining purposes), and the learning must be carried out by keeping minimum information on past training data. Developing a stable one-pass incremental learning algorithm for RBFN will give a great impact to many practical applications.

In order to construct a high-performance classifier/predictor using RBFN, the RBF centers and widths should be determined properly such that the input regions of all training data are fully supported by fewer RBFs. For this purpose, there have been proposed various methods to choose RBF centers and widths in batch learning settings [10,11]. However, as far as we know, there is no online version of such an automated learning algorithm. This is because the distribution of training data is generally unknown in incremental learning environments where the data are only given in a sequential way. Besides, it is well known that the classifier performance could be affected by data preprocessing; especially, it could depend on whether data are normalized or not. As mentioned earlier, in most of the incremental learning algorithms, the parameter setting and the selection of a preprocessing method is conducted by an external supervisor on a trial and error basis when a learning algorithm is applied to a particular problem. However, in incremental learning, this would sometimes be difficult for the supervisor if only a small number of training data are given in the beginning. Therefore, it is important to develop autonomous incremental learning algorithms so that a system can learn on its own without external help [12].

For this purpose, we have proposed an autonomous learning algorithm for RBFN called *Autonomous Learning algorithm for Resource Allocating Network* (AL-RAN) [13]. AL-RAN is a one-pass incremental learning model which consists of the following autonomous functions: 1) data collection for initial learning, 2) data normalization, 3) addition of radial basis functions (RBFs), and 4) determination of RBF centers and widths. The first function enables AL-RAN to decide the necessity of data normalization from incoming training data, and if it is needed, the data scaling is autonomously carried out in an online fashion. And the third and fourth functions allow AL-RAN to free from tuning proper RBF centers and widths before the learning is started. In this AL-RAN model [13], training data are first collected until both the mean and standard deviation are converged. Then, the average recognition accuracies for normalized and unnormalized data are evaluated using the leave-one-out cross-validation method. If the time evolution in recognition accuracy becomes smaller than a threshold, the data collection is stopped and the necessity of data normalization is judged whether the accuracy for unnormalized data is significantly higher than that for normalized data. A problem in this algorithm is that the mean and standard deviation of collected data can largely be fluctuated over time depending on training sequences. Furthermore, evaluating the recognition accuracy at every learning stage would incur large computation costs.

In this paper, to alleviate the above problems, we propose to introduce the class separability instead of the recognition accuracy in a convergence criterion for data collection. The time to stop collecting data is determined by confirming the convergence of the class separability or by finding the significant difference in the class separability for normalized and unnormalized data. In order to determine the initial structure of AL-RAN, a two-stage clustering algorithm is performed for the collected data to obtain a moderate number of prototypes and their cluster radii. These prototypes and radii are set to RBF centers and widths in AL-RAN, respectively. Then, the initial learning of AL-RAN is performed for the collected data to obtain connection weights and memory items in the long-term memory. After the above initial learning process, the learning is switched to the incremental learning phase and the learning is continued forever.

This paper is organized as follows. In Section 2, we first give a brief review of Resource Allocating Network with Long-Term Memory (RAN-LTM) [7,14] which gives a classifier model in AL-RAN, and then a new autonomous learning algorithm for AL-RAN is proposed in Section 3. Then, the performance of AL-RAN is evaluated for the five UCI data sets [15] in Section 4. Finally, conclusions and further work are described in Section 5.

2. Resource Allocating Network with Long-Term Memory

Neural networks often suffer from well-known *catastrophic interference* [9] in incremental learning environments. RAN-LTM [7,14] can alleviate this problem. RAN-LTM consists of Resource Allocating Network (RAN) [8] and Long-Term Memory (LTM). RAN is an extended model of RBFN in which RBFs (*i.e.*, hidden units) are automatically allocated on an online basis. Let

Algorithm 1 Main Program of RAN-LTM

```
Input: Training data $(\boldsymbol{x}, c)$ and RBF width $\sigma$.
 1: Initialize the number of RBF: $J = 1$.
 2: Create an RBF center: $\boldsymbol{\mu}_1 = \boldsymbol{x}$.
 3: Set RBF width: $\sigma_1 = \sigma$.
 4: Set weights $\boldsymbol{W}_1$: $\boldsymbol{W}_1 = \mathbf{1}_c$.
 5: Create a memory item: $(\tilde{\boldsymbol{x}}_1, \tilde{c}_1) = (\boldsymbol{x}, c)$.
 6: loop
 7:    Input: Training data $(\boldsymbol{x}, c)$.
 8:    Calculate the output $\boldsymbol{z}(\boldsymbol{x})$ using Eqs (1) and (2).
 9:    Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \mathbf{1}_c\|/K$.
10:    Find the closest center $\boldsymbol{\mu}_{j'}$ to $\boldsymbol{x}$ and calculate the distance $d^*$.
11:    if $E > 0.5$ and $d^* > \sigma$ then
12:       Perform Add RBF with $(\boldsymbol{x}, c)$, $\boldsymbol{z}(\boldsymbol{x})$, and $\sigma$.
13:       Create a memory item: $(\tilde{\boldsymbol{x}}_J, \tilde{c}_J) = (\boldsymbol{x}, c)$.
14:    else
15:       Perform Learn RBFN with $(\boldsymbol{x}, c)$ and $\{(\tilde{\boldsymbol{x}}_l, \tilde{c}_l)\}_{l=1}^{J}$.
16:       Calculate the output $\boldsymbol{z}(\boldsymbol{x})$ using Eqs (1) and (2).
17:       Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \mathbf{1}_c\|/K$.
18:       if $E > 0.5$ then
19:          Perform Add RBF with $(\boldsymbol{x}, c)$, $\boldsymbol{z}(\boldsymbol{x})$, and $\sigma$.
20:          Create a memory item: $(\tilde{\boldsymbol{x}}_J, \tilde{c}_J) = (\boldsymbol{x}, c)$.
21:       end if
22:    end if
23: end loop
```

Algorithm 2 Add RBF

```
Input: RBFN, training data $(\boldsymbol{x}, c)$, outputs $\boldsymbol{z}(\boldsymbol{x})$, and RBF width $\sigma$.
Output: RBFN.
1: Increment the number of RBFs: $J \leftarrow J+1$.
2: Create an RBF center: $\boldsymbol{\mu}_J = \boldsymbol{x}$.
3: Set RBF width: $\sigma_J = \sigma$.
4: Set weights from the $J$th RBF to outputs: $\boldsymbol{W}_J = \mathbf{1}_c - \boldsymbol{z}(\boldsymbol{x})$.
```

Algorithm 3 Learn RBFN

```
Input: RBFN, training data $(\boldsymbol{x}, c)$, and memory items $\{(\tilde{\boldsymbol{x}}_l, \tilde{c}_l)\}_{l=1}^{J}$.
Output: RBFN.
1: Calculate RBF outputs $\boldsymbol{y}(\boldsymbol{x})$ using Eq. (1).
2: for $l$=1 to $J$ do
3:     Calculate RBF outputs $\boldsymbol{y}(\tilde{\boldsymbol{x}}_l)$ using Eq. (1).
4: end for
5: Define an activation matrix $\boldsymbol{\Phi} = \{\boldsymbol{y}(\boldsymbol{x}), \boldsymbol{y}(\tilde{\boldsymbol{x}}_1), \cdots, \boldsymbol{y}(\tilde{\boldsymbol{x}}_J)\}$.
6: Define a target matrix $\boldsymbol{D} = \{\mathbf{1}_c, \mathbf{1}_{\tilde{c}_1}, \cdots, \mathbf{1}_{\tilde{c}_J}\}$.
7: Decompose $\boldsymbol{\Phi}$ using SVD as follow: $\boldsymbol{\Phi} = \boldsymbol{U}\boldsymbol{H}\boldsymbol{V}'$.
8: Calculate the weights by Eq. (5).
```

us denote the number of inputs, RBFs, and outputs as I, J, K, respectively. The learning algorithm of RAN-LTM is shown in Algorithms 1-3.

Let the inputs of a training data be $\boldsymbol{x}=\{x_1, \ldots, x_I\}'$. Then, the RBF outputs $\mathbf{y}=\{y_1, \ldots, y_J\}'$ are calculated as follow:

$$y_j = \exp\left(-\frac{\|\mathbf{x}-\boldsymbol{\mu}_j\|^2}{\sigma_j^2}\right) \qquad (j=1,\cdots,J) \tag{1}$$

where $\boldsymbol{\mu}_j \in R^I$ and σ_j^2 are the center and the width of the jth RBF. Let the weight from the jth RBF to the kth output be w_{kj} and the bias of the kth output be ξ_k. Then, the outputs $\mathbf{z}(\boldsymbol{x})=\{z_1, \ldots, z_K\}'$ are given by

$$z_k = g\left(\sum_{j=1}^{J} w_{kj} y_j + \xi_k\right) \qquad (k=1,\cdots,K) \tag{2}$$

where

$$g(u) = \begin{cases} 0 & u \le 0 \\ u & 0 < u < 1 \\ 1 & u \ge 1 \end{cases} \tag{3}$$

The data in LTM are called *memory items* that correspond to representative input-output pairs. These pairs are selected from incoming training data, and they are learned with newly given training data to suppress forgetting. As seen in Algorithm 1, a memory item is created when an RBF is allocated; that is, a pair of an RBF center and the corresponding output is stored in LTM as a memory item.

The learning algorithm of RAN-LTM is divided into two phases: the addition of RBFs and the update of connection weights (see Algorithm 1). The procedure in the former phase is the same as that in RAN, except that memory items are created at the same time. Once RBFs are allocated, the centers are fixed afterwards. Therefore, the connection weights $\boldsymbol{W}=\{w_{kj}\}$ are only parameters that are updated based on the output errors. To minimize the errors, the following linear equation is solved [11]:

$$\mathbf{W}'\mathbf{\Phi} = \mathbf{D} \tag{4}$$

where $\boldsymbol{D}$ is the matrix whose column vectors correspond to the targets. Assume that a class c data ($\boldsymbol{x}$, c) is given and J memory items $(\tilde{\mathbf{x}}_l, \tilde{c}_l)$ (l=1,…,J), where $\tilde{c}_l$ is the class label of the lth memory item, are stored in LTM. Then, the target matrix $\boldsymbol{D}$ is formed as follow: $\mathbf{D}=\{\mathbf{1}_c, \mathbf{1}_{\tilde{c}_1}, \cdots, \mathbf{1}_{\tilde{c}_J}\}'$. Here, $\mathbf{1}_c$ is a K-dimensional target vector where the cth element is one and others are zeros. Furthermore, $\mathbf{\Phi}=\{\mathbf{y}(\mathbf{x}), \mathbf{y}(\tilde{\mathbf{x}}_1), \cdots, \mathbf{y}(\tilde{\mathbf{x}}_J)\}$ are defined by the RBF outputs $\boldsymbol{y}$ for the training data and memory items. Using Singular Value Decomposition (SVD), $\boldsymbol{\Phi}$ is decomposed as follow: $\boldsymbol{\Phi}=\boldsymbol{UHV}'$. Then, the weight matrix $\boldsymbol{W}$ in Equation (4) is given by the following equation:

$$\mathbf{W} = \mathbf{U}(\mathbf{H}')^{+}\mathbf{V}'\mathbf{D} \tag{5}$$

where $(\boldsymbol{H}')^{+}$ is the pseudo-inverse of $\boldsymbol{H}'$.

3. Autonomous Incremental Learning

3.1. Assumptions and Learning Scheme

Let us assume that no training data is given to a system in the beginning and training data are provided one by one. Since we expect a system to learn from incoming data with some autonomy, the learning system should collect training data on its own in order to determine an initial network with proper parameters, and then it should be able to improve the recognition accuracy consistently through incremental learning. To earn good performance, an autonomous learning system should judge whether some preprocessing (e.g., data normalization and feature extraction) is needed or not. As mentioned in Section 1, we consider only the data normalization for autonomous preprocessing. After the normalization, collected data are used for constructing and learning an initial AL-RAN classifier (*i.e.*, RAN-LTM). The period up to this point is called *initial learning phase*.

After the initial learning phase, the incremental learning process is evoked. First, the data would be normalized if a system judges that the data normalization is preferable. Then, the incremental learning is conducted not only by updating network weights but also by allocating new RBFs and/or by updating the widths of existing RBFs. This period is called *incremental learning phase*.

3.2. Initial Learning Phase

3.2.1. Autonomous Data Collection and Normalization

In order to judge the necessity of data normalization, an autonomous learning system should collect training data. However, collecting too many training data leads to inefficiency in the computation and memory costs, and it also results in loosing the online learning property because the autonomous system cannot make any prediction until such a large amount of data are collected. Therefore, it is important for a learning system to estimate a sufficient number of data for making a proper decision on the data normalization.

a) Previous Method: In the previous work [13], we proposed a heuristic method to determine the time to stop the data collection based on the following two criteria: the statistics of a data distribution (e.g., mean and standard deviation) and the difference between the recognition accuracies for normalized and unnormalized data (*i.e.*, raw data). First, training data are collected until both the mean and standard deviation are converged. Then, the average recognition accuracy for normalized data R^{nor} and that for raw data R^{raw} are evaluated using the leave-one-out method. If the difference between the two recognition accuracies becomes smaller than a threshold θ_r (*i.e.*, $| R^{\mathrm{nor}} - R^{\mathrm{raw}} | < \theta_r$), the data collection is terminated and the necessity of data normalization is judged; otherwise, more training data are collected until the above condition is satisfied. After satisfying the convergence condition, a system checks the significant difference between R^{raw} and R^{nor}. If R^{nor} is not significantly lower than R^{raw}, the collected data are normalized such that they are subject to a normal distribution N (0,1), and all the data given in future are also normalized in the same way. Otherwise, the data normalization is not applied not only to the collected data but also to training data given in future.

A problem of the above method is that the mean and standard deviation of collected data can largely be fluctuated over time depending on training sequences. If training data are uniformly drawn from a true distribution, a relatively small number of training data would be sufficient to judge the convergence even with the above simple statistics. However, if training data are given based on a biased distribution, the mean and standard deviation of data would converge very slowly. Therefore, relying on only such simple statistics would result in increasing the average number of collected data.

A way to alleviate this problem is to introduce another stopping criterion. Recall that the purpose of the data collection is to judge the necessity of data normalization. Even though the data distribution does not converge, the data collection could be stopped when a system identifies a significant difference in the recognition accuracy for normalized and raw data. A straightforward way to implement this idea is to apply the cross-validation method at every learning stage. Obviously, however, large computation costs are required for this. An alternative way is to adopt the class separability of data to estimate the recognition accuracy.

b) Proposed Method: Assume that n training data $\{(\mathbf{x}_i, c_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i$ and $\boldsymbol{c}_i$, respectively corresponds to the ith input and the class label, are collected so far. Let us further assume that the training data belong to either of C classes and the class c data set is denoted as $\boldsymbol{X}_c=\{\mathbf{x}_{ci}\}_{i=1}^{n_c}$ (c=1,…,C), where $\boldsymbol{x}_{ci} \in R^I$ is the ith data of class c and n_c is the number of class c data. Then, the whole input data set is denoted as $\mathbf{X} = \{\mathbf{X}_c\}_{c=1}^{C}$. For the collected data, the following between-class scatter matrix $\boldsymbol{S}_B$ and the within-class scatter matrix $\boldsymbol{S}_W$ are defined:

$$\mathbf{S}_B = \frac{1}{n}\sum_{c=1}^{C} n_c (\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})' \tag{6}$$

$$\mathbf{S}_W = \frac{1}{n}\sum_{c=1}^{C}\sum_{j=1}^{n_c} (\mathbf{x}_{cj} - \bar{\mathbf{x}}_c)(\mathbf{x}_{cj} - \bar{\mathbf{x}}_c)' \tag{7}$$

where $\bar{\mathbf{x}}_c$ and $\bar{\mathbf{x}}$ are the mean vectors of the class c data and all data, respectively. The class separability P_n for n training data is defined as follow:

$$P_n = \mathrm{tr}\{\mathbf{S}_W^{-1}\mathbf{S}_B\}\,. \tag{8}$$

This class separability P_n can be an alternative measure for the mean and standard deviation adopted in the previous work [13] because P_n includes the mean $\bar{\mathbf{x}}_c$ and the total variance is defined by $\boldsymbol{S}_B$ and $\boldsymbol{S}_W$. Therefore, P_n never converges unless both mean and standard deviation converge.

In this paper, we propose a new heuristic algorithm based on the class separability P_n, which could stop the data collection with a moderate number of training data. Here, let us define the following difference ΔP_n between the class separability of unnormalized (raw) data $\Delta P_n^{\mathrm{raw}}$ and that of normalized data $\Delta P_n^{\mathrm{nor}}$:

$$\Delta P_n = \frac{| P_n^{\mathrm{raw}} - P_n^{\mathrm{nor}} |}{\max\{P_n^{\mathrm{raw}}, P_n^{\mathrm{nor}}\}}\,. \tag{9}$$

As a convergence criterion, we define the following average time-difference for ΔP_n:

$$\delta \bar{P}_n = \frac{1}{\tau}\sum_{k=1}^{\tau} \frac{| \Delta P_{n-k+1} - \Delta P_{n-k} |}{\max\{\Delta P_{n-k+1}, \Delta P_{n-k}\}}\,. \tag{10}$$

In the previous work, the difference between the recognition accuracies for normalized and raw data was adopted. This can be replaced with the difference ΔP_n in Equation (9). Thus, let us define another measure to terminate the data collection as follow:

$$\Delta \bar{P}_n = \frac{1}{\tau}\sum_{k=1}^{\tau} \Delta P_{n-k}\,. \tag{11}$$

Algorithm 4 shows the main procedures of the proposed

Algorithm 4 Autonomous Data Collection

```
Input: Parameters $\theta_s$, $\theta_r$, and $\tau$.
Output: Collected data $\boldsymbol{X} = \{\boldsymbol{X}_c\}_{c=1}^{C}$, data mean $\boldsymbol{m}$, and standard
    deviations $\boldsymbol{\sigma}$.
1:  repeat
2:    Input: Training data $(\boldsymbol{x}, c)$.
3:    if $c$ is a new class then
4:      Increment the number of classes: $C \leftarrow C + 1$.
5:    end if
6:    Add $\boldsymbol{x}$ to class $c$ data set $\boldsymbol{X}_c$.
7:    Increment the numbers of training data and class $c$ data: $n \leftarrow$
      $n + 1$ and $n_c \leftarrow n_c + 1$.
8:    Calculate the class separability $P_n^{\mathrm{raw}}$ for $\boldsymbol{X}$ using Eq. (8).
9:    Normalize $\boldsymbol{X}$ such that they are subject to a normal distribu-
      tion $N(0, 1)$, and denote the normalized data set as $\tilde{\boldsymbol{X}}$.
10:   Calculate the class separability $P_n^{\mathrm{nor}}$ for $\tilde{\boldsymbol{X}}$ using Eq. (8).
11:   Calculate $\delta \bar{P}_n$ using Eq. (10).
12:   if $\delta \bar{P}_n < 2\theta_s$ then
13:     Calculate $\Delta \bar{P}_n$ using Eq. (11).
14:   end if
15: until $\delta \bar{P}_n < \theta_s$ or $\Delta \bar{P}_n > \theta_r$
16: Evaluate the recognition accuracies for raw data $R^{\mathrm{raw}}$ and for
    normalized data $R^{\mathrm{nor}}$ using the leave-one-out cross-validation.
17: if $R^{\mathrm{nor}} \geq R^{\mathrm{raw}}$ then
18:   Calculate the mean $\boldsymbol{m}$ and standard deviations $\boldsymbol{\sigma}$ of $\boldsymbol{X}$.
19:   Normalize $\boldsymbol{X}$ such that they are subject to a normal distribu-
      tion $N(0, 1)$.
20: end if
```

data collection. As seen in Algorithm 4, the class separability is calculated for both raw and normalized data, and the average time-difference δP_n is checked if it is smaller than $2\theta_s$. The reason why a large threshold $2\theta_s$ is used here is to carry out a rough convergence check. After satisfying this condition, the second measure $\Delta \bar{P}_n$ is checked if there is a significant difference between the class separability for normalized and raw data. Therefore, the data collection is terminated not only when the average time-difference of the class separability $\delta \bar{P}_n$ converges but also when the average difference in the class separability for normalized and raw data $\Delta \bar{P}_n$ becomes significant.

After terminating the data collection, the recognition accuracy for raw data R^{raw} and that for normalized data R^{nor} are evaluated using the leave-one-out method. If R^{raw} is significantly higher than R^{nor}, the collected data are used for learning RAN-LTM without normalization, and the data normalization is not also carried out for future training data. Otherwise, the normalized data are used for learning RAN-LTM, and the normalization is always applied to incoming data. Note that the cross-validation is carried out only once. In the following, for a notational convenience, let us express either normalized or raw data by $\mathbf{X}$.

3.2.2. Initial Learning of AL-RAN

Training data $\mathbf{X} = \{\mathbf{X}_c\}_{c=1}^{C} = \{\{x_{ci}\}_{i=1}^{n_c}\}_{c=1}^{C}$ collected in the previous stage are used for the initial learning of AL-RAN, which is divided into the following two processes: determining an initial network structure and computing initial weights.

First, RBF centers are selected from $\boldsymbol{X}$, and then proper RBF widths are determined such that the RBF response fields cover all class regions with a moderate number of RBF centers. It is well known that in order to maximize the generalization performance, the number of RBFs should be as small as possible; therefore, an RBF width should be set as large as possible under the condition that output errors are kept small. For this purpose, we propose an autonomous algorithm (Algorithm 5) to determine RBF centers and widths. The proposed algorithm consists of the following two stages: (1) rough selection of cluster centers (prototype candidates) and (2) selection of prototypes and determination of cluster radii.

In the first stage, the following procedures are carried out for every class c. First, the minimum distance d_{ci}^* between the ith data $\boldsymbol{x}_{ci}$ and other data $\boldsymbol{x}_{cj}$ $(j \neq i)$ is calculated as follow:

$$d_{ci}^{*} = \min_{j \neq i} \left\| \mathbf{x}_{ci} - \mathbf{x}_{cj} \right\|. \tag{12}$$

Then, the minimum distances d_{ci}^* are averaged over all the class c data and the average value $\bar{d}_c$ is calculated. Since $\bar{d}_c$ corresponds to the expected distance to the nearest neighbors within a class c region, the rough selection is performed such that a prototype candidate $\boldsymbol{y}_{cj}$ represents several collected data within a cluster region whose radius should be set larger than $\bar{d}_c$. In order to ensure that at least two data are represented by a prototype in the rough selection, the radius is set to $2\bar{d}_c$ here. As seen in Algorithm 5, the first prototype candidate $\boldsymbol{y}_{c1}$ is randomly selected from the collected data $\boldsymbol{X}_c = \{x_{ci}\}_{i=1}^{n_c}$, and $\boldsymbol{y}_{c1}$ is put into the prototype candidate set $\boldsymbol{Y}_c$. Then, a cluster region is defined such that $\boldsymbol{y}_{c1}$ is set to the center and the radius is set to $2\bar{d}_c$. Then, all the data within the region are represented by $\boldsymbol{y}_{c1}$, and they are removed from $\boldsymbol{X}_c$. The above procedures are continued until $\boldsymbol{X}_c$ becomes empty.

In the second stage, prototype candidates $\{\mathbf{Y}_c\}_{c=1}^{C}$ are further selected to obtain prototypes by merging their cluster regions. As seen in Algorithm 5, the following procedures are carried out for every class c. First, a prototype candidate $\boldsymbol{y}_{ci'}$ is randomly selected, and the distance $d_{ci'}^*$ to the nearest prototype of another class is calculated. Then, $\boldsymbol{y}_{ci'}$ and d_{ci}^* are respectively defined as the first prototype $\boldsymbol{p}_{c1}$ and the cluster radius r_{c1}. And all the data within the cluster are removed from $\boldsymbol{Y}_c$. Such procedures are continued until $\boldsymbol{Y}_c$ gets empty.

At the final stage of the initial learning, the structure of AL-RAN is initialized by creating RBFs and by computing weights. As seen in Algorithm 6, every pair of prototype and the radius $\mathbf{P}_c = \{\{\mathbf{p}_{ci}, r_{ci}\}_{i=1}^{m_c}\}_{c=1}^{C}$ is set to

Algorithm 5 Select Prototypes

```
Input: Collected data $\boldsymbol{X} = \{\boldsymbol{X}_c\}_{c=1}^C = \{\{\boldsymbol{x}_{ci}\}_{i=1}^{n_c}\}_{c=1}^C$.
Output: Prototype $\boldsymbol{P} = \{\boldsymbol{P}_c\}_{c=1}^C = \{\{(\boldsymbol{p}_{c1}, r_{c1})\}_{i=1}^{m_c}\}_{c=1}^C$.
 1: // 1st Stage – Preselection of Prototypes
 2: for $c$=1 to $C$ do
 3:   for $i$=1 to $n_c$ do
 4:     Calculate the minimum distance $d_{ci}^*$ using Eq. (12).
 5:   end for
 6:   Calculate the mean $\bar{d}_c$ of $d_{ci}^*$ $(i = 1, \cdots, n_c)$.
 7:   Select $\boldsymbol{x}_{ci'}$ from $\boldsymbol{X}_c$ at random and $\boldsymbol{X}_c \leftarrow \boldsymbol{X}_c \setminus \boldsymbol{x}_{ci'}$.
 8:   Initialize the number of prototype candidates: $l_c = 1$.
 9:   Put $\boldsymbol{x}_{ci'}$ into the set $\boldsymbol{Y}_c$ as a prototype candidate $\boldsymbol{y}_{c1}$.
10:   for $i$=1 to $n_c - 1$ do
11:     Find the closest candidate in $\boldsymbol{Y}_c$ to $\boldsymbol{x}_{ci}$, and calculate the
        distance $\tilde{d}_{ci}^*$.
12:     if $\tilde{d}_{ci}^* > 2\bar{d}_c$ then
13:       $l_c \leftarrow l_c + 1$ and put $\boldsymbol{x}_{ci}$ into $\boldsymbol{Y}_c$ as $\boldsymbol{y}_{cl_c}$.
14:     end if
15:   end for
16: end for
17: // 2nd Stage – Final Selection of Prototypes
18: for $c$=1 to $C$ do
19:   Initialize the number of prototypes: $m_c = 0$.
20:   repeat
21:     Select a prototype candidate $\boldsymbol{y}_{ci'}$ from $\boldsymbol{Y}_c$ at random,
        $\boldsymbol{Y}_c \leftarrow \boldsymbol{Y}_c \setminus \boldsymbol{y}_{ci'}$, and $m_c \leftarrow m_c + 1$.
22:     Find the closest candidate of a different class to $\boldsymbol{y}_{ci'}$, and
        calculate the distance $d_{ci'}^*$.
23:     Put $(\boldsymbol{y}_{ci'}, d_{ci'}^*)$ into the set $\boldsymbol{P}_c$ as a prototype $(\boldsymbol{p}_{cm_c}, r_{cm_c})$
        where $\boldsymbol{p}_{cm_c}$ and $r_{cm_c}$ correspond to the center and radius
        of $m_c$-th cluster.
24:     for all prototype candidates $\boldsymbol{y}_{cj}$ in $\boldsymbol{Y}_c$ do
25:       Calculate the distance to $\boldsymbol{y}_{ci'}$: $\tilde{d}_{cj} = \|\boldsymbol{y}_{cj} - \boldsymbol{p}_{yi'}\|$.
26:       if $\tilde{d}_{cj} \le d_{ci'}^*$ then
27:         $\boldsymbol{Y}_c \leftarrow \boldsymbol{Y}_c \setminus \boldsymbol{y}_{cj}$.
28:       end if
29:     end for
30:   until $\boldsymbol{Y}_c$ is empty.
31: end for
```

Algorithm 6 Initialize AL-RAN

```
Input: Collected data $\boldsymbol{X} = \{\{\boldsymbol{x}_{ci}\}_{i=1}^{n_c}\}_{c=1}^C$, Prototype $\boldsymbol{P} =$
    $\{\boldsymbol{P}_c\}_{c=1}^C = \{\{(\boldsymbol{p}_{c1}, r_{c1})\}_{i=1}^{m_c}\}_{c=1}^C$.
Output: AL-RAN.
 1: $J$=0
 2: for $c$=1 to $C$ do
 3:   for $j$=1 to $m_c$ do
 4:     Create an RBF center: $\boldsymbol{\mu}_{J+j} = \boldsymbol{p}_{cj}$.
 5:     Set the RBF width: $\sigma_{J+j} = r_{cj}$.
 6:     Calculate the output $\boldsymbol{z}(\boldsymbol{\mu}_j)$ using Eqs (1) and (2).
 7:     Set the weights $\boldsymbol{W}_{J+j}$ between the $(J+j)$th RBF and the
        outputs: $\boldsymbol{W}_{J+j} = \boldsymbol{1}_c - \boldsymbol{z}(\boldsymbol{\mu}_j)$.
 8:     Create a memory item: $(\tilde{\boldsymbol{x}}_{J+i}, \tilde{c}_{J+i}) = (\boldsymbol{\mu}_{J+j}, c)$.
 9:   end for
10:   Increase the number of RBFs: $J \leftarrow J + m_c$.
11: end for
```

RBF center $\boldsymbol{\mu}_j$ and RBF width σ_j (j=1,…, J), respectively. Here, J is the total number of RBFs (*i.e.*, $J = \sum_{c=1}^{C} m_c$). Then, the connection weights $\boldsymbol{W}_j$ from the jth RBF to the outputs are computed as follow: $\boldsymbol{W}_j = \boldsymbol{1}_c - \boldsymbol{z}(\boldsymbol{\mu}_j)$ (j=1,…,J).

After initializing AL-RAN, all of the collected training data $\mathbf{X} = \{\{x_{ci}\}_{i=1}^{n_c}\}_{c=1}^{C}$ are given to AL-RAN one by one, and AL-RAN is learned based on the same incremental learning algorithm described in the next subsection (see Algorithm 7).

3.3. Incremental Learning Phase

In the incremental learning phase, the learning algorithm of AL-RAN is basically the same as that of RAN-LTM [7] except that RBF widths are automatically determined or adjusted in an online fashion. Let us explain how RBF widths are determined from incoming data. In pattern recognition problems, when an RBF center is near a class boundary, the RBF width is generally set to a small value in order to avoid serious overlaps between different classes. On the other hand, when an RBF center is located far from a class boundary, the width should be as large as possible to reduce the number of RBFs. However, since only a part of training data are usually provided at early learning stages, no valid information on class boundaries is generally given; thus, setting too large values to RBF widths could cause serious overlaps to other class regions at later learning stages, resulting in the catastrophic forgetting.

To avoid this, we adopt a safe strategy to determine the width of a newly created RBF. That is, the width of the Jth RBF σ_J is given by the distance to the closest RBF center $\boldsymbol{\mu}_{j^*}$ as follow:

$$\sigma_J = \min_{j \neq J} \left\| \boldsymbol{\mu}_j - \boldsymbol{\mu}_J \right\| \cdot \tag{13}$$

In some cases, the closest RBF may have a large width that seriously overlaps with the support region of the new RBF. Therefore, the width of the closest RBF σ_{j^*} is also resized as follow:

$$\sigma_{j^*} = \min\left(\left\| \boldsymbol{\mu}_J - \boldsymbol{\mu}_{j^*} \right\|, \sigma_{j^*} \right). \tag{14}$$

Algorithm 7 Learn AL-RAN

```
Input: AL-RAN and Training data $(\boldsymbol{x}, c)$.
Output: AL-RAN.
 1: Input $\boldsymbol{x}$ to AL-RAN and calculate the output $\boldsymbol{z}(\boldsymbol{x})$.
 2: Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{1}_c\|/K$.
 3: Find the closest RBF center $\boldsymbol{\mu}_{j^*}$ to $\boldsymbol{x}$ and calculate the distance
    $d^*$.
 4: if $E > 0.5$ and $d^* > \sigma_{j^*}$ then
 5:   Perform Add RBF with $(\boldsymbol{x}, c)$, $\boldsymbol{z}(\boldsymbol{x})$, and $d^*$.
 6:   Create a memory item: $(\tilde{\boldsymbol{x}}_J, \tilde{c}_J) = (\boldsymbol{x}, c)$.
 7:   Resize the $j^*$th RBF width: $\sigma_{j^*} \leftarrow \min(\sigma_{j^*}, d^*)$
 8: else
 9:   Perform Learn RBFN to learn AL-RAN using $(\boldsymbol{x}, c)$ and
      $\{(\tilde{\boldsymbol{x}}_l, \tilde{c}_l)\}_{l=1}^J$.
10:   Input $\boldsymbol{x}$ into RAN-LTM to calculate the output $\boldsymbol{z}(\boldsymbol{x})$.
11:   Calculate the error: $E = \|\boldsymbol{z}(\boldsymbol{x}) - \boldsymbol{1}_c\|/K$.
12:   if $E > 0.5$ then
13:     Perform Add RBF with $(\boldsymbol{x}, c)$, $\boldsymbol{z}(\boldsymbol{x})$, and $d^*$.
14:     Create a memory item: $(\tilde{\boldsymbol{x}}_J, \tilde{c}_J) = (\boldsymbol{x}, c)$.
15:     Resize the $j^*$th RBF width: $\sigma_{j^*} \leftarrow \min(\sigma_{j^*}, d^*)$
16:   end if
17: end if
```

The incremental learning algorithm of AL-RAN is summarized in Algorithm 7.

3.4. Summary of Learning Algorithm and Intrinsic Parameters

The learning algorithm of AL-RAN is summarized in Algorithm 8. The initial learning phase is shown in Lines 2-5, and the remaining part corresponds to the incremental learning phase that could be repeated forever. Therefore, we can say that the proposed AL-RAN model is categorized into a lifelong learning model.

Here, it should be noted that the inputs of Algorithm 8 are the parameters θ_s, θ_r, and τ that are used in the autonomous data collection (see Algorithm 4). Although these parameters should be determined in advance, they are essentially different from so-called network parameters such as the number of RBFs and RBF widths which should be properly set depending on given problems. The above parameters mainly determine the behavior of an autonomous learning system.

As described in Section 3.2.1, the threshold θ_s gives an upper bound to judge the convergence of the class separability; thus, if θ_s is set small, a strict convergence criterion is applied, resulting in collecting many training data to make a decision. On the other hand, the threshold θ_r gives a lower bound of the separability difference between normalized and raw data. Therefore, if θ_r is set small, this leads to a loose convergence criterion for the data collection. The parameter τ determines the period to monitor the degree of satisfying the convergence criteria. If τ is set small, a nearsighted convergence decision would be made, resulting in a loose convergence criterion. Hence, we can say that these parameters determine the behavioral property of an autonomous learning system. Let us call θ_s, θ_r, and τ *intrinsic parameters*.

In the current version of AL-RAN, such intrinsic parameters should be set in advance by an external supervisor. In this sense, the proposed AL-RAN is not a fully autonomous learning model. However, these parameters are relatively independent of given problems because normalized convergence criterions are adopted (see Equations (9-11)). Therefore, it is not difficult to set them properly.

Algorithm 8 Main Program of AL-RAN

```
Input: Parameters $\theta_s$, $\theta_r$, and $\tau$.
 1: // Initial Learning Phase
 2: Perform Autonomous Data Collection.
 3: Perform Select Prototypes with collected data $\boldsymbol{X}$.
 4: Perform Initialize AL-RAN with $\boldsymbol{X}$ and the prototypes $\boldsymbol{P}$.
 5: for all collected data $(\boldsymbol{x}_i, c_i)$ in $\boldsymbol{X}$ do
 6:     Perform Learn AL-RAN with $(\boldsymbol{x}_i, c_i)$.
 7: end for
 8: // Incremental Learning Phase
 9: loop
10:     Input: Training data $(\boldsymbol{x}, c)$.
11:     if data normalization is selected then
12:         Normalize $\boldsymbol{x}$ with mean $\boldsymbol{m}$ and standard deviations $\boldsymbol{\sigma}$.
13:     end if
14:     Perform Learn AL-RAN with $(\boldsymbol{x}, c)$.
15: end loop
```

3.5. Discussions on Learning Convergence

Let us discuss the learning stability of AL-RAN by dividing the learning algorithm into the three parts: 1) the initial learning phase (Algorithms 4-6), 2) incremental learning of individual training data (Algorithm 7), and 3) the main flow of the incremental learning phase (Algorithm 8).

As seen in Algorithm 4, the autonomous data collection process would not stop if the two conditions at Line 15 are not satisfied forever. That is to say, the data collection could continue forever if the class separability of collected data does not converge or significant difference between the class separabilities for normalized and raw data is not detected. This can happen when the threshold θ_s is too small and θ_r is set too large. These thresholds are called intrinsic parameters whose values are supposed to be set properly by an external supervisor. As long as these parameters are set properly, the data collection would stop after a finite number of data are collected. However, it might not be easy even for a supervisor to set it properly under actual environments where the distribution of training data is dynamically changed over time. Therefore, at the current version of AL-RAN, a perfect convergence for the initial learning phase is not theoretically guaranteed. However, as will be demonstrated later, the data collection always stops in our experiments (see **Table 3**). If the data collection stops, it means that the number of collected data is finite. Then, the other parts in the initial learning phase would terminate with finite operations (see Algorithms 5 and 6).

As seen in Algorithm 7, the learning algorithm of AL-RAN does not include iterative calculations that can continue forever. The weights are obtained with the matrix computation in Equation (5). In the augmentation of hidden units (RBFs), at most one hidden unit is created per training data. Therefore, the convergence of incremental learning for individual training data is ensured from the algorithmic point of view.

Since we assume general incremental learning environments, the main routine of the incremental learning phase would continue forever. If we restrict the learning under a stationary environment, it would be possible to show a mathematical proof for the learning convergence. However, we assume not only stationary environments but also nonstationary environments where the data distribution could be dynamically changed over time. Therefore, we cannot discuss learning stability for the main

Table 1. Evaluated UCI Data sets. "Norm." means the preference of data normalization: + means that data should be normalized.

Data Set	#Attrib	#Class	#Train	#Test	Norm.
Vowel-context	10	11	485	485	—
Vehicle	18	4	423	423	+
Segmentation	10	11	1,155	1,155	+
Thyroid	21	3	3,600	3,600	+
Breast-cancer	31	2	284	285	+

Table 2. The number of collected data and decision accuracy [%] for data normalization.

(a) Vowel-context

			θ_s			
			0.005	0.01	0.02	0.05
θ_r	0.1	#Collected Data	64.6±15.7	43.2±11.7	31.2±7.1	22.8±5.0
		Accuracy [%]	76	70	64	42
	0.25	#Collected Data	65.8±15.2	47.1±13.7	34.0±9.0	25.9±6.5
		Accuracy [%]	80	66	64	46
	0.5	#Collected Data	99.0±25.0	64.2±16.2	43.2±11.7	27.4±6.3
		Accuracy [%]	84	76	70	48
	1.0	#Collected Data	99.0±25.0	64.6±15.7	43.2±11.7	27.4±6.3
		Accuracy [%]	84	76	70	48

(b) Thyroid

			θ_s			
			0.005	0.01	0.02	0.05
θ_r	0.1	#Collected Data	39.2±19.6	25.9±11.7	17.4±7.3	12.8±5.2
		Accuracy [%]	88	94	88	84
	0.25	#Collected Data	39.2±19.6	25.9±11.7	17.6±7.8	12.8±5.3
		Accuracy [%]	88	94	88	84
	0.5	#Collected Data	41.4±22.7	29.7±18.1	19.9±9.9	13.9±6.0
		Accuracy [%]	92	96	90	84
	1.0	#Collected Data	62.7±29.5	39.2±19.6	25.9±11.7	15.4±5.9
		Accuracy [%]	90	88	94	82

routine of the incremental learning phase under such general learning environments.

4. Experiments

4.1. Experimental Setup

Five data sets are selected from UCI Machine Learning Repository [15] to evaluate the performance of AL-RAN. **Table 1** shows the information on the data sets. Although training and test data are separately provided in some data, they are merged and randomly divided into two subsets to conduct the two-fold cross-validation. Since the performance generally depends on the sequence of cross-validation round (*i.e.*, fifty sequences in total) are trained and the average performance is evaluated for the test data. We assume that training data are given one by one in a random order. The column ``Norm." in **Table 1** shows the preference of data normalization: + means that the recognition accuracy is higher when the data is normalized. This preference was preliminary examined by evaluating the test performance of an RBF network which is learned with all the training data in a batch mode. Note that this preference is not informed to AL-RAN.

The performance evaluation of AL-RAN is conducted through the comparison with the previous AL-RAN [13] and RAN-LTM. Let us denote the new and previous AL-RANs as AL-RAN(new) and AL-RAN(old), respectively. Here, we adopt the following performance measures: 1) decision accuracy for data normalization, 2) the number of collected training data, 3) learning time, 4) the number of RBFs, and 5) final test recognition accuracy. The first measure is adopted to evaluate the autonomous data collection and the autonomous decision of data normalization in the initial learning phase. The decision accuracy is defined as the rate that the decision of the data normalization is matched with the preference in **Table 1**. Needless to say, the performance of the initial learning phase would be better if a higher decision accuracy is obtained by collecting a smaller number of training data. The second through fourth scales are adopted to evaluate the efficiency in terms of learning time and network structure. The fifth scale is adopted to evaluate the total performance of the proposed autonomous learning scheme.

Table 3. Performance evaluations for (a) the proposed AL-RAN, (b) the previous AL-RAN, and (c) non-autonomous learning model RAN-LTM. The two values in each cell are an average value and the standard deviation in the form of $x \pm y$.

(a) AL-RAN(new)

	Decision Accur. [%]	#Collected Train. Data	Time [sec.] for Init. Learning	Time [sec.] for Incre. Learning	#RBFs at Final Learning Stage	Final Recog. Accur. [%]
Vowel-context	76	64.2±16.2	0.34±0.14	12.0±2.0	179.3±8.3	96.8±0.9
Vehicle	82	77.9±37.6	0.51±0.42	7.8±1.3	177.9±11.4	75.0±3.1
Segmentation	70	113.2±61.1	0.92±0.84	21.2±4.4	166.9±14.9	94.2±0.8
Thyroid	96	29.7±18.1	0.21±0.09	431.6±382.3	336.5±112.0	93.6±2.8
Breast-cancer	98	21.4±10.1	0.07±0.04	0.31±0.25	38.1±12.1	94.4±2.4

(b) AL-RAN(old)

	Decision Accur. [%]	#Collected Train. Data	Time [sec.] for Init. Learning	Time [sec.] for Incre. Learning	#RBFs at Final Learning Stage	Final Recog. Accur. [%]
Vowel-context	82	91.4±9.2	1.13±0.48	12.8±1.8	186.7±9.0	97.1±0.9
Vehicle	94	72.8±11.3	0.77±0.36	8.7±1.2	185.4±9.1	75.9±2.5
Segmentation	78	82.0±11.3	0.98±0.50	31.4±9.1	191.2±21.5	94.2±0.9
Thyroid	90	85.0±13.1	1.36±1.83	728.6±624.4	387.6±103.7	94.1±2.7
Breast-cancer	100	40.8±10.1	0.31±0.13	0.50±0.19	47.5±7.3	95.2±1.4

(c) RAN-LTM

	Decision Accur. [%]	#Collected Train. Data	Time [sec.] for Init. Learning	Time [sec.] for Incre. Learning	#RBFs at Final Learning Stage	Final Recog. Accur. [%]
Vowel-context	–	–	–	25.1±2.1	203.9±5.7	98.1±0.6
Vehicle	–	–	–	9.8±1.3	161.5±5.1	76.3±1.8
Segmentation	–	–	–	115.5±11.0	290.4±5.5	94.2±0.8
Thyroid	–	–	–	3877.1±563.7	653.0±25.2	91.8±1.4
Breast-cancer	–	–	–	2.25±0.17	113.6±3.4	96.2±1.0

4.2. Effects of Intrinsic Parameters

Before evaluating the performance of AL-RAN, let us examine how the intrinsic parameters affect the behavior of AL-RAN. As described in Section 3-4, τ and θ_r have a similar effect on the model behavior. Hence, we only study the influence of θ_s and θ_r for simplicity. In the following experiments, τ is fixed at 5.

Tables 2 (a) and **(b)** show the number of collected data and the decision accuracy in the initial learning phase for (a) Vowel-context data and (b) Thyroid data. As seen in **Tables 2 (a)** and **(b)**, when θ_s is getting small, the number of collected data is increasing and the decision accuracy for data normalization becomes high. The same is true when θ_r is getting large. Although this is not a surprising result, an interesting interpretation is that these intrinsic parameters control the discretion of a system. That is, a system with a smaller θ_s would be more discreet to make a decision, while a system with a smaller θ_r would become more optimistic.

4.3. Performance Evaluation

Table 3 shows the decision accuracy for data normalization, the number of collected data in the initial learning phase, the time required for initial learning and incremental learning, the number of RBFs, and the test recognition accuracy after all the learning is completed. The parameters used in the experiments are $\theta_r = 0.5$ and $\theta_s = 0.01$. For a comparative purpose, we also evaluate two incremental learning models. The one is the previous AL-RAN model (AL-RAN(old)) which adopts different convergence criteria on the data collection and a different method to initialize AL-RAN in the initial learning phase (see [13] for details). The other is RAN-LTM in which a fixed RBF width is given and the data normalization is determined based on the preference shown in **Table 1**; that is, the learning of RAN-LTM is not autonomously carried out.

As seen in **Tables 3 (a)** and **(b)**, the number of collected training data in AL-RAN(new) is significantly decreased as compared with AL-RAN(old) except for Vehicle and Segmentation data. If the number of collected data is small, the incremental learning could start early; that is, the proposed AL-RAN(new) can start classification and incremental learning at earlier learning stages than the previous model. The time needed for the initial learning of the proposed AL-RAN(new) is also significantly shorter than that of AL-RAN(old). Even though the number of collected data is increased in AL-RAN(new) for Vehicle and Segmentation data, the time for initial leaning is shortened. This is because the convergence for data collection is checked based on the class separability whose computation costs are smaller than those for the cross-validation in AL-RAN(old).

On the other hand, the decision accuracy for data normalization is slightly degraded in AL-RAN(new). Although this may cause different performance degradation, as long as we can see the results in **Table 3 (a)** and **(b)**, the test accuracy of AL-RAN(new) is not degraded significantly after all incremental learning is completed. This means that the learning of AL-RAN(new) can compensate the misjudgment in the data normalization process; therefore, the degradation in the decision accuracy does not harm seriously to the final test performance. The above results also suggest that the online learning property of the proposed AL-RAN(new) is improved without sacrificing the test recognition performance.

As seen in **Tables 3 (a)** and **(b)**, the number of RBFs in AL-RAN(new) is decreased as compared with AL-RAN(old). This result mainly comes from the difference in how to construct an initial AL-RAN. Since a two-stage clustering is applied to selecting RBF centers and widths in AL-RAN(new), a smaller number of RBFs are selected after the initial learning phase. This results in a compact structure in AL-RAN(new) and contributes to fast incremental learning. As seen in **Table 3 (c)**, the number of RBFs in RAN-LTM could be significantly larger than that of AL-RAN depending on the data sets. This is because the RBF width is fixed with a predetermined value[1] even for an RBF whose center is located far from decision boundaries. Therefore, we can say that an autonomous determination of RBF widths works well.

5. Conclusions

In this paper, we propose a new version of the autonomous incremental learning algorithm called AL-RAN. AL-RAN consists of the autonomous data normalization and the autonomous scaling of RBF widths which enable a system to learn without external human intervention. There are three intrinsic parameters θ_s, θ_r, and τ in the convergence criteria, which should be determined by an external supervisor. However, it should be noted that these thresholds are independent of given problems because normalized convergence criteria are adopted in the data collection. In the current version of AL-RAN, these parameters should be determined from the aspect of what type of autonomous systems the supervisor wants to create. An interesting approach to making AL-RAN fully autonomous is to introduce an evolutionary computation algorithm to find optimal intrinsic parameters.

[1] The RBF width was determined such that RAN-LTM has similar recognition accuracy against AL-RAN.

The experimental results for the five UCI data sets demonstrate that although the decision accuracy for data normalization is relatively lower than the previous AL-RAN model, the number of collected data is decreased and the time for the initial learning is also shorten. From the fact that the test recognition accuracy of the proposed AL-RAN is not degraded significantly after all incremental learning is completed, the degradation in the decision accuracy for the data normalization is not a critical problem. Therefore, we can conclude that the proposed AL-RAN can estimate a moderate number of training data to conduct an efficient learning. Furthermore, the introduction of a two-stage clustering for selecting RBF centers and widths results in a smaller network structure, which contributes to fast incremental learning. Finally, the recognition performance of AL-RAN is fairly good compared with RAN-LTM, in which proper network parameters and the data normalization method are determined by an external supervisor. From the experimental results, we can conclude that the autonomous learning in AL-RAN works very well.

There still remain several open problems in AL-RAN. Although the robustness of the convergence condition in the data collection is improved by considering not only the convergence on the average time-difference of the class separability but also the significant difference in the class separability for normalized and raw data, there still be no guarantee to stop the data collection at the moderate number of data. If training data are given based on a strongly biased distribution, many training data might be collected, and then a learning system would have to wait for long time until incremental learning is started. Therefore, it might be effective to change the thresholds θ_s and θ_r (see the 15th line in Algorithm 4) over time during the initial learning phase. The proposed learning algorithm works in real time if the dimensions of data are not too high. For high-dimensional data (e.g., image data and DNA microarray data), however, the real-time learning might not be ensured. To overcome this problem, a dimensional reduction technique (e.g., feature selection and extraction methods) is usually adopted as preprocessing. For this purpose, we have developed incremental version of LDA [16] and PCA [7]. Therefore, in order to ensure real-time learning, it is useful to combine such incremental feature extraction methods with the proposed AL-RAN for high-dimensional data. Since the decision on data normalization is not perfect (see **Table 3**), it is better to switch to a preferable normalization method even during the incremental learning phase. However, there is one problem to do that. Since the knowledge is encoded in connection weights and RBF centers, it is not easy to learn new knowledge with keeping old knowledge if the current and past data are transformed with different normalization methods. If this problem is solved, it is expected that the idea of online normalization would work well to improve the final classification accuracy. Finally, the current version of AL-RAN cannot handle missing values. This function is also solicited for a real autonomous learning algorithm. The above problems are left as our future work.

6. Acknowledgements

The authors would like to thank Prof. Shigeo Abe for his helpful comments. This research was supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C) 20500205.

REFERENCES

[1] R. Sun and T. Peterson, "Autonomous Learning of Sequential Tasks: Experiments and Analyses," *IEEE Transaction on Neural Networks*, Vol. 9, No. 6, 1998, pp. 1217-1234.

[2] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur and E. Thelen, "Autonomous Mental Development by Robots and Animals," *Science*, Vol. 291, 2001, pp. 599-600.

[3] X. Song, C.-Y. Lin and M.-T. Sun, "Autonomous Learning of Visual Concept Models," *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. 5, 2005, pp. 4598-4601.

[4] B. Bolder, H. Brandl, M. Heracles, H. Janssen, I. Mikhailova, J. Schmudderich and C. Goerick, "Expectation-Driven Autonomous Learning and Interaction System," *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, Korea, 2008, pp. 553-560.

[5] P. M. Roth and H. Bischof, "Active Sampling via Tracking," *IEEE-CS Conference on Computer Vision and Pattern Recognition Workshop*, Anchorage, 2008, pp. 1-8.

[6] J.-X. Peng, K. Li and G. W. Irwin, "A Novel Continuous Forward Algorithm for RBF Neural Modeling," *IEEE Transactions on Automatic Control*, Vol. 52, No. 1, 2007, pp. 117-122.

[7] S. Ozawa, S.-L. Toh, S. Abe, S. Pang and N. Kasabov, "Incremental Learning of Feature Space and Classifier for Face Recognition," *Neural Networks*, Vol. 18, No. 5-6, 2005, pp. 575-584.

[8] J. Platt, "A Resource-allocating Network for Function Interpolation," *Neural Computation*, Vol. 3, No. 2, 1991, pp. 213-225.

[9] N. Kasabov, "Evolving Connectionist Systems: Methods and Applications in Bioinformatics, Brain Study and Intelligent Machines," Springer, New York, 2002.

[10] A. Roy, S. Govil and R. Miranda, "An Algorithm to Generate Radial Basis Function (RBF)-Like Nets for Classification Problems," *Neural Networks*, Vol. 8, No. 2, 1995, pp. 179-202.

[11] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice-Hall, USA, 1999.

[12] A. Roy, "Connectionism, Controllers and a Brain Theory," *IEEE Transactions on System, Man and Cybernetics*, *Part A*, Vol. 38, No. 6, 2008, pp. 1434-1441.

[13] T. Tabuchi, S. Ozawa and A. Roy, "An Autonomous Learning Algorithm of Resource Allocating Network," In: E. Corchado and H. Yin, Eds., *Intelligent Data Engineering and Automated Learning—IDEAL* 2009, *LNCS*, Springer, New York, October 2009, Vol. 5788, pp. 134-141.

[14] K. Okamoto, S. Ozawa and S. Abe, "A Fast Incremental Learning Algorithm of RBF Networks with Long-Term Memory," *Proceedings of International Joint Conference on Neural Networks*, USA, 2003, pp. 102-107.

[15] A. Asuncion and D. J. Newman, "UCI Machine Learning Repository," *UC, Irvine, School of Information and Computer Science*, California, 2007.

[16] S. Pang, S. Ozawa and N. Kasabov, "Incremental Linear Discriminant Analysis for Classification of Data Streams," *IEEE Transactions on Systems, Man, and Cybernetics*, *Part B*, Vol. 35, No. 5, 2005, pp. 905-914.