Incremental Learning with Respect to New Incoming Input Attributes

Sheng-Uei Guan* and Shanchun Li

Department of Electrical and Computer Engineering, National University of Singapore

10 Kent Ridge Crescent, Singapore 119260

Abstract

Neural networks are generally exposed to a dynamic environment where the training patterns or the input attributes (features) will likely be introduced into the current domain incrementally. This paper considers the situation where a new set of input attributes must be considered and added into the existing neural network. The conventional method is to discard the existing network and redesign one from scratch. This approach wastes the old knowledge and the previous effort. In order to reduce computational time, improve generalization accuracy, and enhance intelligence of the learned models, we present *ILIA* algorithms (namely ILIA1, ILIA2, ILIA3, ILIA4 and ILIA5) capable of *Incremental Learning in terms of Input Attributes*. Using the ILIA algorithms, when new input attributes are introduced into the original problem, the existing neural network can be retained and a new sub-network is constructed and trained incrementally. The new sub-network and the old one are merged later to form a new network for the changed problem. In addition, ILIA algorithms have the ability to decide whether the new incoming input attributes are relevant to the output and consistent with the existing input attributes or not and suggest to accept or reject them. Experimental results show that the

* Corresponding author. E-mail address: eleguans@nus.edu.sg

ILIA algorithms are efficient and effective both for the classification and regression problems.

Keywords: Constructive learning algorithms, Incremental learning, Input attributes, Neural networks, Supervised learning

1. Introduction

Multilayered feedforward neural networks are widely used to realize nonlinear mappings between the input space and output space with a set of parameters. During the learning phase, the networks basically are assumed to exist in a static environment, i.e., the dimension of the input space is fixed and all sets of training patterns are provided initially. The networks adapt themselves to the static environment by updating their parameters and sometimes by updating their structures (network size) as well (e.g., constructive learning algorithms [1]). However, in reality, the networks are generally exposed to a dynamic environment instead of a static one. Such changing factors in the dynamic environment can be classified into two situations:

- The sets of training patterns are not provided completely in the initial state. During the training process, new incoming information, corresponding to new training patterns are found and introduced into the current system.
- During the training phase, a set of new input attributes (features) is introduced into the current system, corresponding to increasing the dimension of the input space.

In such a dynamic environment, a neural network needs to learn the patterns or input attributes incrementally. Many researchers have presented incremental learning methods for the first situation. Fu et al. [2] proposed an incremental method for pattern recognition, called “incremental backpropagation learning network”, which employs bounded weight modification and structural adaptation learning rules and applies initial knowledge to constrain the learning process. Bruzzon et al. [3] proposed a novel classifier based on RBF neural networks for remote-sensing images. Hebert et al. [4] proposed a method to combine an unsupervised self-organizing map with a multilayerd feedforward neural network to form the hybrid Self-Organizing Perceptron network for character detection. These methods can adapt network structure and/or parameters to learn new incoming patterns automatically, without forgetting previous knowledge. For these methods, we can see *incremental learning* is defined, with respect to training patterns, as a process of updating a previously trained network to learn a set of new incoming training patterns acquired after the network is trained without repeating the entire design and training procedure using the complete training sets.

In this paper, we consider the second situation where a new set of input attributes must be considered and added into the current system. We assume the original problem has N_1 input values (an input attribute has one or more input values) and K output values as shown in Figure 1. Now another set of input values needs to be considered and added into the problem domain and the new input vector consists of N input values. It is possible that the combined inputs may require a larger network than the existing network. The conventional solution is to discard the existing network and redesign a new network based on the new input vector. However, this approach wastes the previous training effort. Normally, this is not a problem because training is done off-line. There are situations where one might want to train the network incrementally in real-time without discarding the existing network, if it can be done quickly and the performance obtained is acceptable, especially when the existing inputs and the training set are large and the time available is short. Applications can be found like on-line stock forecasting systems where new factors affecting stock markets have come to scene, critical medical diagnostic systems where new factors contributing to a disease in concern have been found and re-training needs to be done on the fly, real-time command control systems where new factors affecting situation analysis have come into effect and no training time can be wasted, etc.

In this paper, we will focus on *Incremental Learning in terms of Input Attributes* (ILIA) and the corresponding algorithms as ILIA algorithms. ILIA algorithms construct and train a new sub-network using the added input attributes based on the existing network. They have the ability to train incrementally and allow the system to modify the existing network without excessive computation. The ILIA algorithms are presented in details in section 2 and the corresponding experiments and their results are illustrated in section 3. Section 4 is the concluding remark.

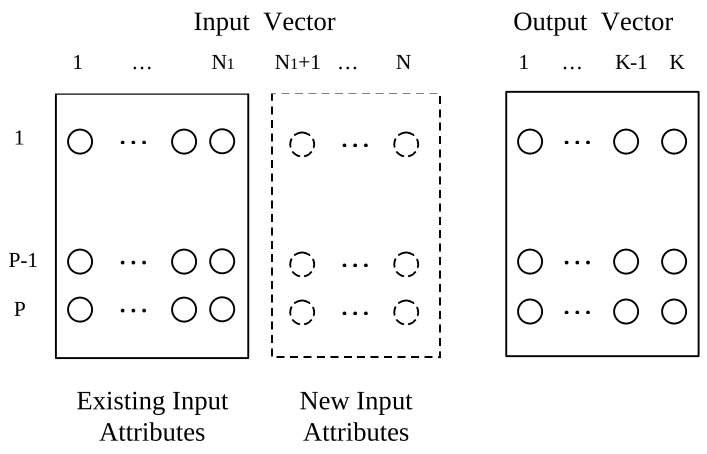


Figure 1. New attributes are introduced into the input vector

2. ILIA Algorithms

2.1 Design Goals

In order to reduce excessive computation and increase learning speed, improve generalization accuracy, and enhance intelligence of the learned models, the proposed ILIA algorithms should meet the following design goals.

Design goal 1: The neural network must automatically grow to an appropriate size without excessive computation.

It is widely known that network architecture is of crucial importance for neural networks. Too small a network cannot learn the problem well [5], while a size too large will lead to overfitting and thus poor generalization [6]. So it is a key issue in neural network design to find appropriate network architecture automatically and optimize the set of weights for the architecture. In order to fulfill the first design goal, constructive neural networks are used. These networks are able to construct their own topology according to some pre-defined criteria without prior knowledge. As a result, the previously trained network is extended to incorporate constructive algorithms with incremental learning capability. Our key idea is to grow another sub-network that is dependent on the previous trained network after a set of new input attributes is introduced into the current domain. The additional training process begins with a small neural network. The network grows in size constructively only if needed. This approach is expected to be more efficient than the network pruning approach that begins with a large network [7].

Design goal 2: The network architecture and training approach must allow incremental learning. And the ILIA algorithms should have the ability to determine whether to accept or reject new incoming input attributes.

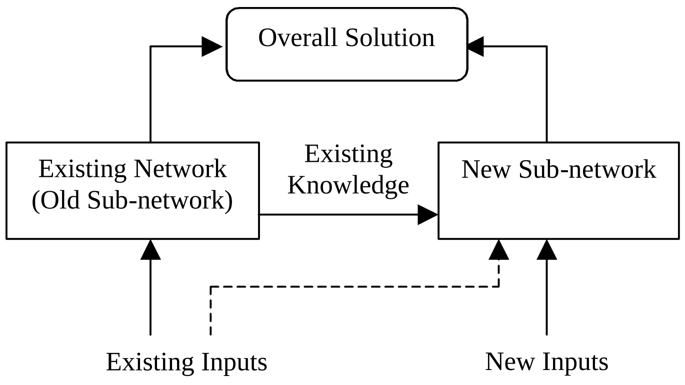
The network should adopt some methods to adapt to the new environment, because learning should be a continuous and incremental process. The ILIA algorithms should have the ability to determine whether the new input attributes are relevant (important) to the output and consistent with the existing input attributes or not. Relevant and consistent input attributes, if used for further network training, can bring about gradual improvement in network performance. In contrast, detrimental effects will result from irrelevant or inconsistent input attributes.

Design goal 3: Learning new knowledge without forgetting existing knowledge. The existing knowledge should be preserved after each incremental learning step.

Existing knowledge should not be wasted. Instead, some way should be devised for an incremental network to exploit the existing knowledge effectively. In addition, the existing knowledge can be exploited during the incremental learning process, especially if the new incoming input attributes are found to be relevant and consistent with the existing input attributes. This way, learning time can be reduced.

2.2 Procedure for the ILIA Algorithms

For ILIA algorithms, a changing environment is defined as a problem with new incoming input attributes, i.e., after a neural network is trained, the input dimension that is equal to the number of input units can increase when new input attributes arrive. Therefore, the number of input units has to be increased. In our ILIA algorithms, there is no need to reconstruct the whole network when the environment is changing. It suffices to increase the number of input units and grow a new portion of neural network corresponding to the new incoming input attributes based on the existing knowledge (network). The main difficulty is how to use the existing knowledge in incremental learning. In this paper, we tried five different algorithms, namely, ILIA1, ILIA2, ILIA3, ILIA4, and ILIA5. The overall scheme of the ILIA algorithms is shown in Figure 2.



. **Figure 2. The overall scheme of the ILIA algorithms**

ILIA algorithms are composed of three stages. The procedure for ILIA1 is as follows.

In stage 1, the existing network is retained as the old sub-network, as shown in Figure 2 and Figure 3 (a). Present all the training, validation, and test patterns to the old sub-network. The sum of weighted inputs for each output unit on each pattern is calculated and stored in arrays.

In stage 2, grow and train the new sub-network.

Step 1: Expand the input dimension. Add input units to the new sub-network to reflect the expansion of the dimension of the input space. The newly added ($N - N_1$) input values consist of the inputs of the new sub-network; and the sum of weighted inputs from the old sub-network and new sub-network goes to the input of each output unit, as shown in Figure 3 (b).

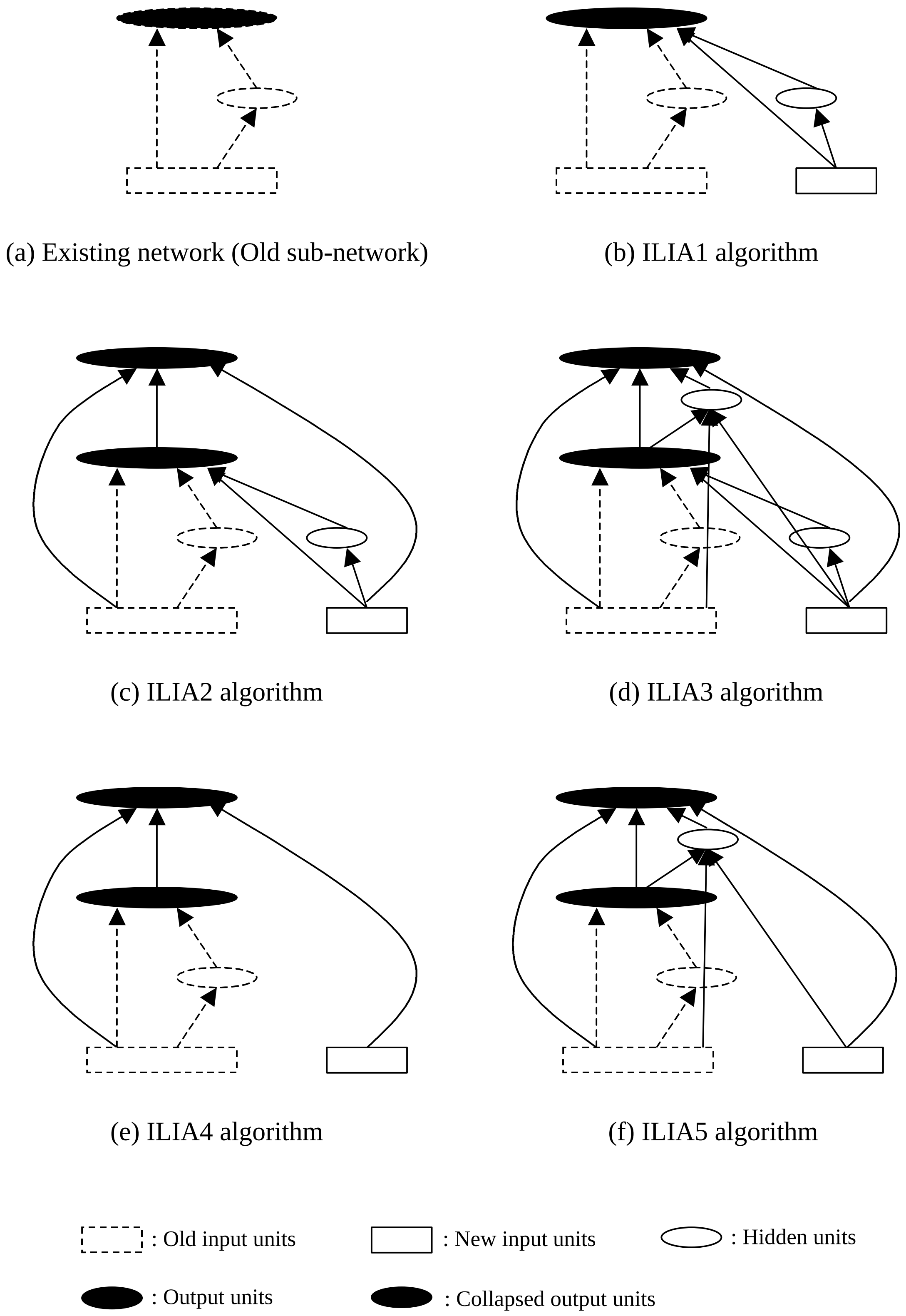


Figure 3. The network structure for the ILIA algorithms

Step 2: The new input units are connected to the output units of the original network. A two-layer feedforward network is obtained.

Step 3: Train the two-layer network obtained in step2. During the process of adjusting the input weights of each output unit, only the weights from the new sub-network are adjusted. The sum of weighted inputs for each output unit on each pattern is the sum of weighted inputs from the old sub-network and the new sub-network.

Step 4: Generate a pool of candidate units and select the best candidate unit from the pool and install it into the new sub-network. Each new hidden unit is connected to all the output units and the new input units, as shown in Figure 3 (b). During the training process, only the weights connected to the new hidden unit are adjusted and all the previously installed units (and weights) are all fixed. This way train and install hidden units as many as possible until the overall stopping criteria are satisfied.

In stage 3, determine whether the new incoming input attributes are relevant to the output and consistent with the existing input attributes or not. If the new incoming input attributes are relevant to the output and consistent with the existing input attributes, then merge the new sub-network with the old sub-network to form the new neural network (overall solution) for the changed problem. Otherwise, reject the new incoming input attributes and retain the old sub-network.

The differences between the procedure for the other ILIA algorithms and that for ILIA1 lie in stag 2. Stage 1 and 3 are the same for all of them. For ILIA2 and ILIA3, the major differences from ILIA1 are as follows:

- For ILIA2, in stage 2, based on the structure grown and trained from ILIA1, collapse the output layer as shown in Figure 3 (c). This means that the K units that were output units so far are now deemed to be the hidden units in the next hidden layer. Add the new output units and connect them to all the input units and the collapsed output layer (now it is a hidden layer). Then adjust all the newly added connections.
- For ILIA3, in stage 2, based on the structure obtained in ILIA2, train and install hidden units. Generate a pool of candidate units and select the best candidate unit from the pool and install it into the new sub-network. As shown in Figure 3 (d), the new hidden unit receives input connections from the collapsed output layer and all the input units. This way train and install hidden units as many as possible until the overall stopping criteria are satisfied.

The stage 2 for ILIA4 is as follows:

Step 1: Expand the input dimension. Add input units to the new sub-network to reflect the expansion of the dimension of the input space.

Step 2: Collapses the output layer of the existing network (old sub-network) directly in stage 2, as shown in Figure 3 (e). Add the new output units and connect them to all the input units and the collapsed output layer (now it is a hidden layer).

Step 3: Train the newly added connections. During the training process, the old sub-network is fixed and only the weights from the new sub-network, i.e. newly added connection are adjusted.

For ILIA5, the major difference from ILIA4 is the following. In stage 2, based on the structure obtained in ILIA4, ILIA5 generates a pool of candidate units and selects the best candidate unit from the pool and installs it into the new sub-network. As shown in Figure 3 (f), the new hidden unit receives input connections from the collapsed output layer and all the input units. This way ILIA5 trains and installs hidden units as many as possible until the overall stopping criteria are satisfied.

An intuitive explanation of our approaches - and why they work- is the following. When the problem has changed with a new set of incoming input attributes, the old sub-network previously trained can still be used in constructing the new neural net, as the existing input attributes are still valid. For ILIA1, with the new input attributes, a new sub-network is grown with the purpose to decide for each output unit the effect from the new input attributes. So it is grown solely based on the new input set. This new sub-network is later merged with the old sub-network to superimpose their effects on each output unit. This should be achievable from our assumption that the new input attributes have no clash with the existing input attributes. If the new input attributes are not consistent with the old ones, we can use some detecting mechanism to detect and reject them, which will be explained in details in section 3.2.1. For ILIA2, based on the network obtained by ILIA1, the original output layer is collapsed to become a hidden layer. This way, ILIA2 can obtain more information than ILIA1. Firstly, it collapses the original output layer, therefore, it has the potential to grasp the higher-order information and has the chances to "update" or "improve" the existing network via the added connections between the new output units and the collapsed output units. Secondly, the new output units are fed the connections from all the input units (including the existing input units and new ones) at the same time. For ILIA3, based on the network obtained by ILIA2, it continues to install hidden units into the new sub-network. The motivation is to obtain more information than ILIA2. ILIA4 collapses the output layer of the old sub-network directly after a new set of input units are added. The motivation of ILIA4 is to gain the higher-order information of the old sub-network and the information from all the inputs together by collapsing the output layer. Based on ILIA4, ILIA5 continues to add the hidden units.

2.3 Some Definitions and the Stopping Criteria for Growing and Training the Sub-network

As mentioned in the previous sections, constructive learning algorithms are incorporated into the ILIA algorithms. There are many constructive learning algorithms, such as the *Constructive Backpropagation* (CBP) algorithm [8], *Cascade-Correlation* (CC) algorithm [9], *Dynamic Node Creation* (DNC) method [10], and *Tiling* algorithm [11], etc. In this paper, we adopt the CBP algorithm. The reason why CBP is selected is that the implementation of CBP is simple and we do not need to switch between two different cost functions like in the CC algorithm. And we only need to backpropagate the output error through one and only one hidden layer. This way the CBP algorithm is computationally as efficient as the CC algorithm [8].

Although constructive learning algorithms have many advantages [1, 12], they are very sensitive to changes in the stopping criteria. If training is too short, the components of the network will not work well to generate good results. If training is too long, it costs much computation time and may result in overfitting and poor generalization. Referring to [13, 14], we adopted the method of *early stopping* using a validation set to prevent overfitting.

The set of available patterns is divided into three sets: a *training set* is used to train the network, a *validation set* is used to evaluate the quality of the network during training and to measure overfitting, and a *test set* is used at the end of training to evaluate the resultant network. The size of the training, validation, and test set is 50%, 25% and 25% of the problem's total available patterns.

The error measure E used is *the squared error percentage* [13], derived from the normalization of the mean squared error to reduce the dependency on the number of coefficients in the problem representation and on the range of output values used:

$$E = 100 \cdot \frac{o_{\max} - o_{\min}}{K \cdot P} \sum_{p=1}^{P} \sum_{k=1}^{K} (o_{pk} - t_{pk})^2$$

where $o_{\max}$ and $o_{\min}$ are the maximum and minimum values of output coefficients in the problem representation.

$E_{tr}(t)$ is the average error per pattern of the network over the training set, measured after epoch t. The value $E_{va}(t)$ is the corresponding error on the validation set after epoch t and is used by the stopping criterion. $E_{te}(t)$ is the corresponding error on the test set; it is not known to the training algorithm but characterizes the quality of the network resulting from training.

The value $E_{opt}(t)$ is defined to be the lowest validation set error obtained in epochs up to epoch t:

$$E_{opt}(t) = \min_{t' \le t} E_{va}(t')$$

The *generalization loss* [13] at epoch t is defined as the relative increase of the validation error over the minimum so far (in percent):

$$GL(t) = 100 \cdot (\frac{E_{va}(t)}{E_{opt}(t)} - 1)$$

A high generalization loss is one candidate reason to stop training because it directly indicates overfitting.

To formalize the notion of training progress, a *training strip of length* k [13] is defined to be a sequence of k epochs numbered $n+1 \ldots n+k$ where n is divisible by k. The training *progress* measured after a training strip is:

$$P_k(t) = 1000 \cdot (\frac{\sum_{t' \in t-k+1 \ldots t} E_{tr}(t')}{k \cdot \min_{t' \in t-k+1 \ldots t} E_{tr}(t')} - 1)$$

It is used to measure how much larger the average training error is than the minimum training error during the training strip.

During the process of growing and training sub-networks, we adopted the following heuristic overall stopping criteria: $E_{opt} < E_{th}$ **OR** *(Reduction of training set error due to the last new hidden unit is less than 0.01%* **AND** *Validation set error increased due to the last new hidden unit).* The first part ($E_{opt} < E_{th}$) means that the optimal validation set error is below the threshold and the result has been acceptable. The other part means the last insertion of a hidden unit resulted in hardly any progress. The criteria for adding a new hidden unit are as follows: *At least 25 epochs reached for the current network* **AND** *(Generalization loss $GL(t)$ >5* ***OR*** *Training progress $P_k(t)$ <0.1).* The first part means that the current network should be trained for at least a certain number of epochs before a new hidden unit is installed because the error curves will be turbulent in the beginning.

The second part means that the current network has been overfitted or training has little progress.

3. Experimental Results and Analysis

3.1 The Experiment Scheme

We have run a number of benchmark problems to evaluate our proposed ILIA algorithms. In order to simulate the arrival of new input attributes, the training patterns of the benchmark problems were partitioned into two independent portions. The first portion was used to grow and train the old sub-network. The obtained old sub-network was regarded as the existing network. The other portion was considered as the new input attributes and the new sub-network was constructed using the ILIA algorithms. In this paper, we will report the results of two classification problems (*Diabetes1*, and *Thyroid1* problem) and one regression problem (*Flare1* problem). These three benchmark problems are all taken from the PROBEN1 benchmark collection [13] and they all are real-world problems.

In the sets of experiments undertaken, we ran 20 trials with each algorithm for each problem. The RPROP algorithm [15] was used to minimize the cost function. The RPROP algorithm used the following parameters: $h^+ = 1.2$, $h^- = 0.5$, $\Delta_0 = 0.1$, $\Delta_{\max} = 50$, $\Delta_{\min} = 1.0e-6$, with initial weights from –0.25 … 0.25 randomly. In all experiments, 8 candidates were adopted and E_{th} was set to 0.1. The hidden units and output units all used the sigmoid activation function. All the experiments were conducted using a Pentium III – 650 PC.

3.2 Results and Analysis

Several issues are of particular importance: generalization accuracy, learning speed, and the network complexity. As to generalization accuracy, for the classification problems, we pay more attention to classification error than test error; for the regression problems, we pay more attention to test error. It should be noted that the number of old input units, the number of new input units, and the number of total input units are different. Therefore, the computational cost of one epoch can differ significantly between the old sub-network, new sub-network, and the network obtained using the conventional method. Comparing the number of epochs solely will be misleading. So for learning speed, we place the emphasis on training time instead of epochs. As far as network complexity is concerned, the total number of independent parameters (the number of weights and biases in the net) is more significant than the total number of hidden units due to the same reason.

3.2.1 Diabetes1

The Diabetes1 problem diagnoses diabetes of Pima Indians. It has 8 inputs (8 attributes), 2 outputs and 768 patterns. All inputs are continuous. Its attributes are: number of times pregnant, plasma glucose concentration, diastolic blood pressure, triceps skin fold thickness, 2-hour serum insulin, body mass index, diabetes pedigree function, and age.

We conducted 3 groups of experiments for the Diabetes1 problem. In the first group of experiments, we selected 7 input attributes for training the old sub-network and assumed the remaining 1 input attribute as the new incoming input attribute. The results obtained are displayed in Figure 4 - 7. In the X axis, "attribute x" means that all input attributes except the x-th attribute are assumed as the existing attributes and the x-th attribute is assumed as the new incoming input attribute. For example, for attribute 1 (the first column), we assumed all input attributes except the 1st attribute as the existing attributes and assume the 1st input attribute as the new incoming attribute. An "old sub-network" was constructed for the existing input attributes and its classification error is 23.13%, as shown in Figure 4. After the 1st input attribute was introduced into the system, ILIA1, ILIA2, ILIA3, ILIA4 and ILIA5 were used to construct the new sub-networks and their classification errors are 23.28%, 23.41%, 23.54%, 23.18% and 24.26% respectively. For this case, it can be seen that the new incoming attribute has negative effect (larger classification error) on the old sub-network's performance. However, for other cases, e.g. when attribute 2 instead is introduced into the system, ILIA1, ILIA2, ILIA3, ILIA4 and ILIA5 all incur much less classification error (23.67%, 24.51%, 23.10%, 24.78% and 24.26% respectively) than the old sub-network (its classification error is 30.99%).

We can divide the input attributes into the following two classes:

- Class1: The old sub-network's classification error is relatively large. After the new incoming input attribute (e.g. attributes 2 and/or 6) is introduced into the problem domain, the classification error will be significantly reduced by using any ILIA algorithm.
- Class2: The old sub-network's classification error is relatively small. After the new incoming input attribute is introduced into the problem domain, the classification error is increased by using any ILIA algorithm or the classification error is increased by some ILIA algorithms and reduced by other ILIA algorithms, e.g. attribute 1, 3, 4, 5, 7 and 8.

What are the main reasons causing the input attributes to have different performances? Input attributes have different relevance/importance for a problem [16, 17]. If an input attribute is relevant to the problem, then introducing this new incoming input attribute can enhance the performance (reducing the classification error). On the other hand, if a new incoming input attribute is irrelevant to the outputs or inconsistent with the previous ones, it will have little or negative effect on the old sub-network's performance. It should be mentioned that in our ILIA algorithms, we assume that existing attributes are all relevant. Therefore, for Class1 attributes, they are relevant and important input attributes. Note that a feature (input attributes) selector presented in [16] depicted that the Diabetes1 problem has about 2.03 relevant features (input attributes). Our results are consistent with theirs as it shows only attributes 2 and 6 are relevant. In contrast, Class2 attributes are irrelevant ones or they are inconsistent with the existing attributes.

From Figure 4 - 5, we can see that ILIA algorithms can improve performance with Class1 input attributes. Using attribute 2 as an example, the old sub-network's classification error is 30.99%. All ILIA algorithms reduced the classification error significantly and ILIA1 and ILIA3 have smaller classification error than those obtained by the other ILIA algorithms and the conventional method. For attribute 6, all ILIA algorithms have smaller classification error than that obtained by the conventional method. In addition, ILIA algorithms can detect Class2 input attributes. To deal with Class2 attributes, ILIA algorithms can reject them and retain the old sub-network. It can be seen, for attributes 1, 3, 7 and 8, the old sub-network's classification error is smaller than those obtained by ILIA algorithms. For attributes 4 and 5, although ILIA3 and ILIA5 have smaller errors than the old sub-network, the differences are negligible and retaining the old sub-network can also obtain smaller classification error than the conventional method. The final classification errors obtained by ILIA using this detecting mechanism are displayed in Figure 5. We will elaborate the comparison of these ILIA algorithms to the conventional method in terms of training time and the number of independent parameters shortly later.

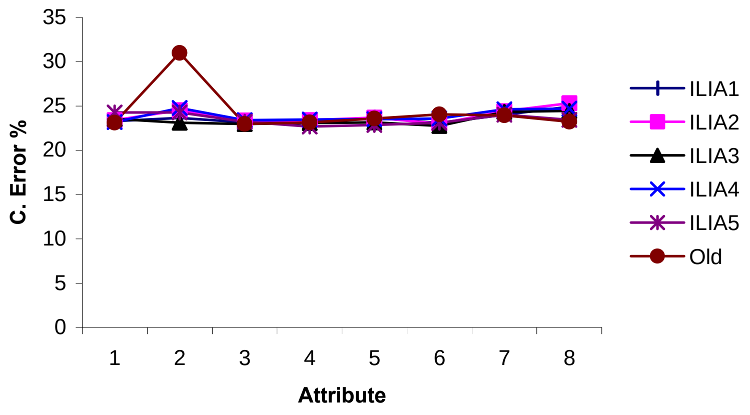


Figure 4. Classification errors of Diabetes1 before using the class-detecting mechanism (Note: “Old” stands for the old sub-network.)

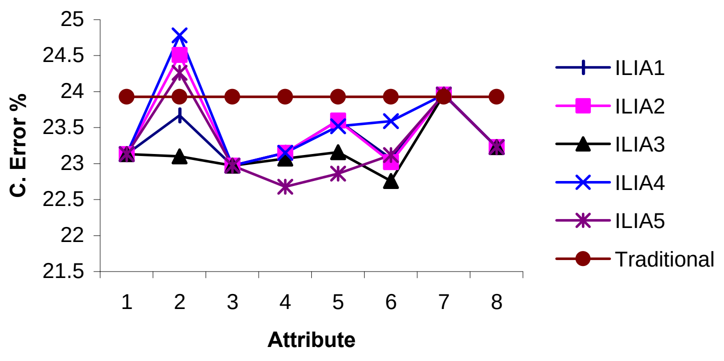


Figure 5. Classification errors of Diabetes1 after using the class-detecting mechanism (for a Class1 attribute, select ILIA’s result; for a Class2 attribute, select the old ~~neural~~ sub-network’s result)

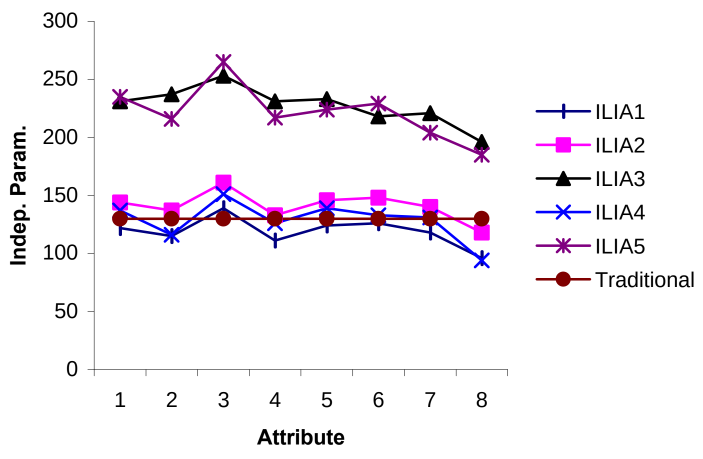


Figure 6. Number of i~~I~~ndependent parameters of Diabetes1 (Note: The number of independent parameters obtained by an ILIA algorithm is the sum of those from the old sub-network and the new sub-network.)

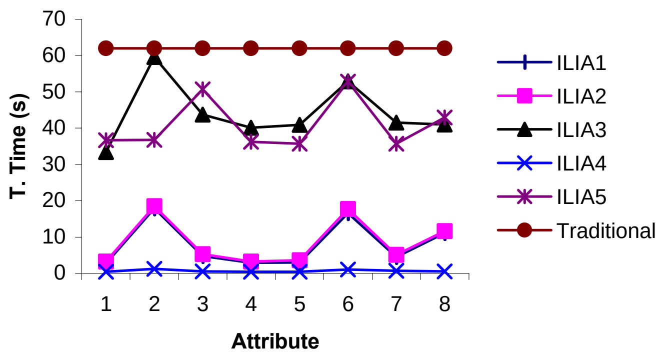


Figure 7. Training time of Diabetes1 (Note: Training time spent by an ILIA algorithm, or say, to construct a new sub-network, ~~are~~ is independent of the old sub-network, i.e. we need not to include the training time spent by the old sub-network because it is an existing network.)

In the second group of experiments, we selected 6 input attributes for training the old sub-network and assumed the remaining 2 input attributes as the new incoming input attributes. The results are shown in Table 1. In the third group of experiments, we selected 4 input attributes for training the old sub-network and assumed the remaining 4 input attributes as the new incoming input attributes. The results are shown in Table 2.

From Table 1 and 2, we can see that attributes 2 and 6 belong to Class1 again, indicating these two attributes are really important attributes.

Table 1. Results for Diabetes1 (Second Group of Experiments)

Problem		Epochs	T. Time (s)	Hidden Units	Indep. Param.	Ete	C. Error (%)
Diabetes1 (Conventional method)		8870	62.0	10.2	130	16.12	23.93
Attributes 1, 2	Old sub-network	5204	34.2	8.1	87	19.74	32.47
	ILIA1	3876	20.7	14.6	121	15.96	22.89
	ILIA2	3957	21.3	16.6	143	16.06	24.77
	ILIA3	8074	52.9	23.4	231	15.57	23.26
	ILIA4	147	1.1	10.1	109	16.57	26.17
	ILIA5	3938	31.0	16.9	197	16.08	24.51
Attributes 3, 4	Old sub-network	7258	47.9	10.2	106	15.59	22.92
	ILIA1	806	4.0	12.0	117	15.61	22.79
	ILIA2	878	4.5	14.0	139	15.95	23.52
	ILIA3	6544	47.8	21.6	238	15.89	22.99
	ILIA4	75	0.6	12.2	128	15.93	23.52
	ILIA5	5399	42.4	19.7	226	15.91	23.28
Attributes 5, 6	Old sub-network	6890	45.3	10.4	108	16.89	24.61
	ILIA1	4712	25.2	14.6	129	16.26	23.44
	ILIA2	4848	26.2	16.6	151	16.02	22.79
	ILIA3	9965	75.3	22.6	229	15.93	22.99
	ILIA4	142	1.1	12.4	130	16.31	23.15
	ILIA5	5593	43.9	18.4	208	16.16	22.53
Attributes 7, 8	Old sub-network	4513	29.6	5.9	67	16.68	23.91
	ILIA1	2553	13.6	10.9	91	16.33	24.51
	ILIA2	2604	14.1	12.9	113	16.91	24.71
	ILIA3	8533	59.5	19.9	204	16.26	23.98
	ILIA4	56	0.6	7.9	89	16.91	25.39
	ILIA5	5554	43.7	14.1	170	16.51	23.59

Table 2. Results for Diabetes1 (Third Group of Experiments)

Problem		Epochs	T. Time (s)	Hidden Units	Indep. Param.	Ete	C. Error (%)
Diabetes1 (Conventional method)		8870	62.0	10.2	130	16.12	23.93
Attributes 1, 2, 5, 6	Old sub-network	3619	21.5	8.4	90	21.42	34.65
	ILIA1	5175	31.2	14.7	119	16.80	23.22
	ILIA2	5260	31.9	16.7	141	16.42	23.57
	ILIA3	9886	67.4	24.0	236	16.14	23.05
	ILIA4	227	1.7	10.4	112	16.95	23.46
	ILIA5	5038	39.1	15.8	182	16.54	23.57
Attributes 3, 4, 7, 8	Old sub-network	3810	22.4	5.7	65	16.08	23.44
	ILIA1	5382	32.1	11.7	97	16.08	23.93
	ILIA2	5440	32.5	13.7	119	16.70	24.92
	ILIA3	11710	80.3	20.9	213	16.04	23.20
	ILIA4	68	0.7	7.7	87	16.59	24.58
	ILIA5	8113	63.5	16.6	203	16.31	24.01

From Figure 5~~—7~~ and Table 2, we can see that ILIA3 has the smallest classification errors in most cases. It is also noted that ILIA3 and ILIA5 resulted in more independent parameters and hidden units and spent comparable training time ~~compared with~~as the conventional method. In Table 1, ILIA1 and ILIA2 have the smallest classification errors. ILIA4 spent the least training time. However, on the whole, ILIA1 and ILIA2 obtain satisfactory classification error compared with the conventional method and they spent much less training time than the conventional method. What's more, the number of independent parameters and hidden units obtained by ILIA1 and ILIA2 are always comparable ~~with~~ to the conventional method.

3.2.2 Thyroid1

Thyroid1 diagnoses whether a patient's thyroid has overfunction, normal function, or underfunction based on patient query data and patient examination data. Thyroid1 has 21 inputs (21 attributes), 3 outputs, and 7200 patterns.

From Figure 8, we see that most input attributes (i.e., except attributes 10~12) belong to Class1, especially for attributes 16~18. In the case of "attributes 16~18", we can see the old sub-network's classification error is 6.44%. When attributes 16~18 were introduced into the system, the classification error was reduced significantly (ILIA1: 2.19%, ILIA2: 1.56%, ILIA3: 1.46%, ILIA4: 2.38%, ILIA5: 2.24%). In the case of "attributes 10~12", however, the old sub-network's classification error is 1.79% and it was increased after the new attributes were introduced into the system. Therefore, there is one or more attributes among attributes 10~12 that belongs to Class2, resulting in negative effect on the old sub-network.

The final classification errors, **number of** independent parameters, and training time are displayed in Figure 9 – 11.We see that ILIA3 obtained the smallest classification error in most cases. However, it usually spent more training time and resulted in a larger number of independent parameters than the conventional method. Compared with ILIA3, ILIA5 also obtained a relatively complex network architecture. It is noted that ILIA2 also obtained a smaller classification error than the conventional method. In addition, compared with the conventional method, ILIA2 spent much less training time and has a comparable number of independent parameters. Compared with ILIA2, ILIA4 usually spent ~~the least~~less time but its classification errors are greater than ILIA2's.

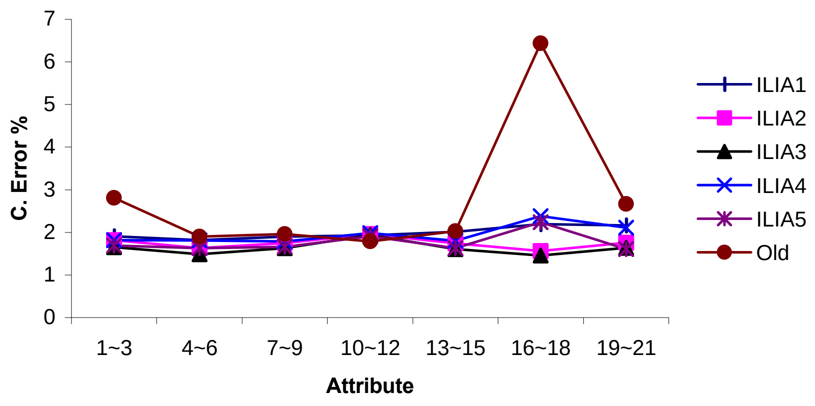


Figure 8. Classification errors of Thyroid1 before using the class-detecting mechanism

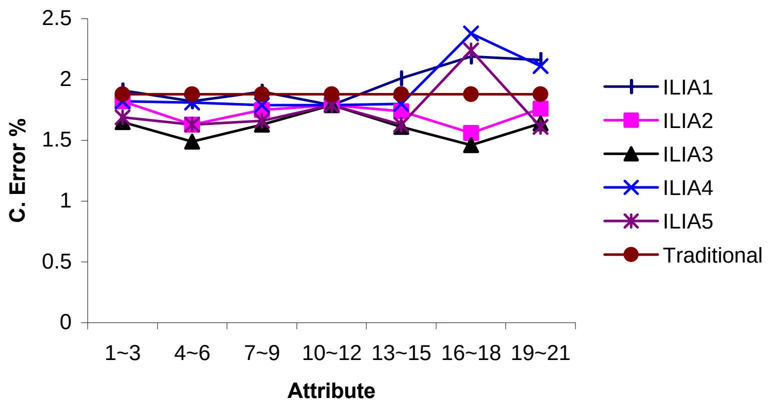


Figure 9. Classification errors of Thyroid1 after using the class-**detecting mechanism**

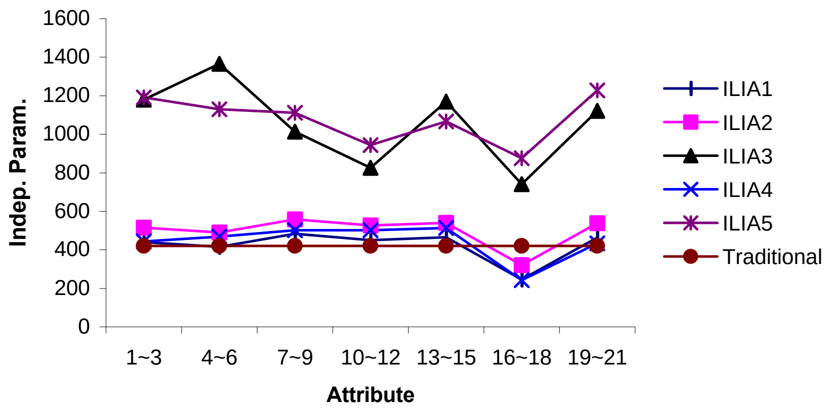


Figure 10. Number of i~~I~~ndependent parameters of Thyroid1

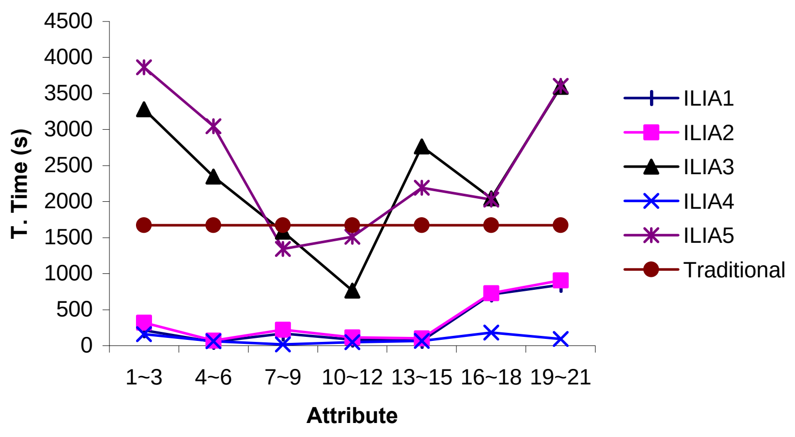


Figure 11. Training time of Thyroid1

3.2.3 Flare1

Flare1 is a regression problem. It predicts solar flares by trying to guess the number of solar flares of small, medium, and large sizes that will happen during the next 24-hour period in a fixed active region of the Sun surface. Its input values describe previous flare activity and the type and history of the active region. Flare1 has 24 inputs (10 attributes), 3 outputs, and 1066 patterns.

For the Flare1 problem, we conducted two groups of experiments. The results show that most attributes belong to Class1 and they are relevant attributes, especially for attribute 3. From Figure 12 – 15 (first group of experiments), we see that ILIA1 reduced the test error for some cases and keep the test error unchanged for the other cases. ILIA2 obtained the smallest test error. Similar to the previous two problems, ILIA2 spent much less training time (17.7 ~ 62.5s, in average 31.4s) than the conventional method (226.1s). At the same time, it obtained a smaller number of independent parameters than the conventional method. ILIA3 and ILIA5 increased the test error since it introduced relatively too many independent parameters and hidden units and tended to be overfitting.

Although ILIA4 reduced the test error and spent least training time, the test error obtained is generally larger than ILIA2. Therefore, ILIA2 is better than the other ILIA algorithms.

From Table 3 (second group of experiments), we can see that attributes 1~3 are more important than attributes 4~10, which is consistent with the results obtained in the first group of experiments. ILIA2 and ILIA4 obtained the same test error. Although ILIA4 spent less training time, ILIA2's results are still acceptable.

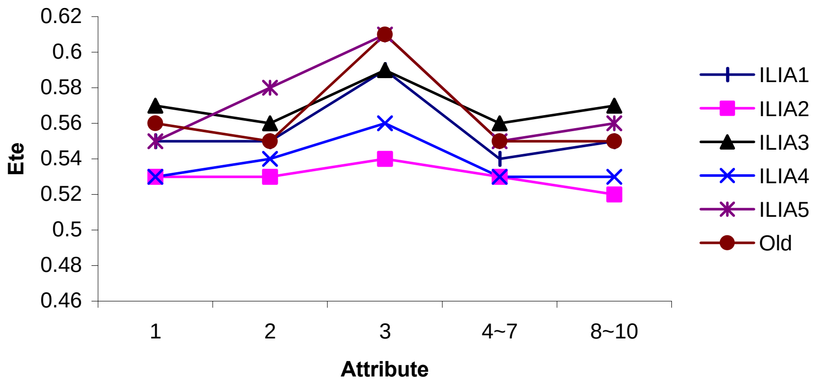


Figure 12. Test errors of Flare1 before using the class-detecting mechanism

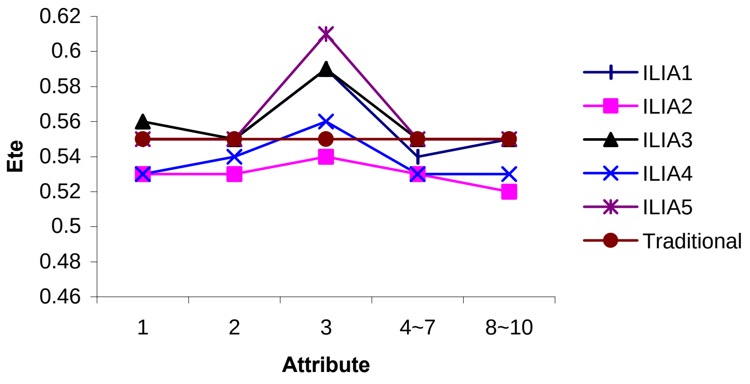


Figure 13. Test errors of Flare1 after using the class-detecting mechanism

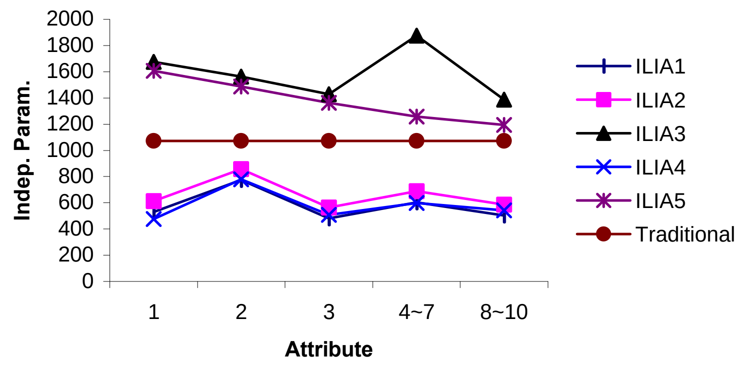


Figure 14. Number of i~~I~~ndependent parameters of Flare1

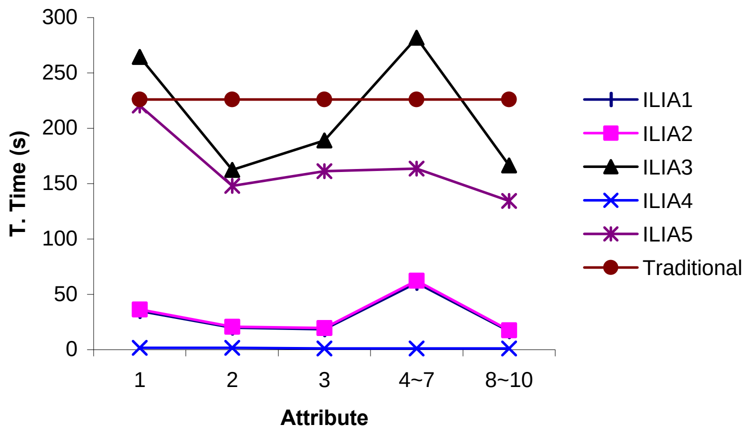


Figure 15. Training time of Flare1

Table 3. Results for Flare1 (Second Group of Experiments)

Problem		Epochs	T. Time (s)	Hidden Units	Indep. Param.	Ete
Flare1 (Conventional method)		11174	226.1	35.6	1072	0.55
Attributes 1~ 3 (3~~17~~ inputs)	Old sub-network	5416	70.7	18.7	230	0.60
	ILIA1	5546	99.0	21.8	717	0.58
	ILIA2	5593	100.6	24.8	801	0.53
	ILIA3	14604	298.6	51.6	1632	0.55
	ILIA4	68	2.6	21.7	314	0.53
	ILIA5	7751	170.2	47.4	1111	0.55
Attributes 4 ~ 10 (7 inputs)	Old sub-network	5148	89.5	18.1	434	0.55
	ILIA1	6330	80.6	11.4	569	0.55
	ILIA2	6409	83.4	14.4	653	0.54
	ILIA3	15910	292.3	43.4	1552	0.56
	ILIA4	43	1.5	21.1	518	0.54
	ILIA5	9164	190.1	54.5	1553	0.57

4. Concluding Remark

Incremental learning is a desirable feature that eliminates the need to redesign and retrain a network from scratch. The ILIA algorithms proposed in this paper are for incremental learning in terms of new input attributes particularly. Using these algorithms, when a set of new input attributes is introduced into the current problem, the network previously trained can be retained as the old sub-network and a new sub-network is constructed and trained. The new sub-network and the old one are merged later to form a new solution for the changed problem.

ILIA1 retains the existing network as an old sub-network and constructs a new sub-network based on the new input attributes. Based on ILIA1, ILIA2 collapses the original output layer to become a hidden layer. ILIA3 continues to install hidden units into the new sub-network based on that obtained from ILIA2. Although ILIA1 grasps the new incoming input attributes basically, ILIA2 can obtain more information than ILIA1. This is because: firstly, it collapses the original output layer, therefore, it has the potential to grasp the higher-order information and has the chances to "update" or "improve" the existing network via the added connections between the new output units and the collapsed output units; secondly, the new output units are fed the connections from all the input units (including the existing input units and new ones) at the same time. ILIA3, based on the network obtained by ILIA2, continues to install hidden units into the new sub-network. Although ILIA3 has the potential to obtain more information than ILIA2, ILIA3 tends to result in too many independent parameters and more training time. For some problems having small number of training patterns, too many independent parameters will cause overfitting, which is reflected in the Flare1 problem. Unlike ILIA2, ILIA4 collapses the original output layer directly after a new set of input units are added. Based on ILIA4, ILIA5 continues to add the hidden units. From the results, we can see that ILIA4 usually spends less training time than ILIA2 while the latter obtains better generalization accuracy than the former. Like ILIA3, ILIA5 tends to result in too many independent parameters and more training time.

On the whole, by using the ILIA algorithms existing knowledge can be preserved instead of being discarded. In addition, the ILIA algorithms have the ability to decide whether the new incoming input attributes are relevant to the outputs or not and suggest to accept or reject them. Consequently, a neural network can be grown incrementally to an appropriate size without excessive computation. In general, the ILIA algorithms can obtain better generalization ability and spend much less training time (with ILIA1, ILIA2, and ILIA4) than the conventional method. At the same time, the obtained network complexity (with ILIA1, ILIA2, and ILIA4) is comparable to or less than that of the conventional method. Generally speaking, ILIA2 algorithm is better than the other ILIA algorithms.

In this paper, we assume that only one set of new input attributes is introduced into the current system. Actually the ILIA algorithms can also be extended smoothly to continuous incremental learning. That means we can introduce new input attributes more than once. It should be mentioned that there exist many variations of the ILIA algorithms proposed in this paper, for example, we can collapse the output layer more times other than only once as in our ILIA algorithms (ILIA2, ILIA3, ILIA4, and ILIA5). These variations will be considered further in our future work.

References

[1] T. Y. Kwok and D. Y. Yeung, “Objective functions for training new hidden units in constructive neural networks”, IEEE Transactions on Neural Networks, Vol. 8, pp. 1131-1148, 1997.

[2] L. M. Fu, H. -H. Hsu and J. C. Principe, “Incremental backpropagation learning networks”, IEEE Transactions on Neural Networks, Vol. 7, pp. 757-761, 1996.

[3] L. Bruzzon and P. D. Fernandez, “An incremental-learning neural network for the classification of remote - sensing images”, Pattern Recognition Letters, Vol. 20, pp. 1241-1248, 1999.

[4] J. -F. Hebert, M. Parizeau and N. Ghazzali, “Cursive character detection using incremental learning”, in Proceedings of the Fifth International Conference on Document Analysis and Recognition, pp. 808 – 811, 1999.

[5] A. Blum and R. L. Rivest, “Training a 3-node neural network is NP-complete”, Neural Networks, Vol. 5, pp. 117-128, 1992.

[6] E. B. Baum and D. Haussler, “What size net gives valid generalization?” Neural Computation, Vol. 1, pp. 151-160, 1989.

[7] R. Reed, “Pruning algorithm—a survey”, IEEE Transactions on Neural Networks, Vol. 4, pp. 740-747, 1993.

[8] M. Lehtokangas, “Modelling with constructive backpropagation”, Neural Networks, Vol. 12, pp. 707-716, 1999.

[9] S. E. Fahlman and C. Lebiere, “The cascade-correlation learning architecture”, in D. S. Touretzky (eds) Advances in neural information processing systems, Vol. 2, pp. 524-532, Morgan Kaufmann Publishers, CA, 1990.

[10] T. Ash, “Dynamic node creation in backpropagation networks”, Connection Science, Vol. 1, pp. 365-375, 1989.

[11] M. Mezard and J. P. Nadal, “Learning in feedforward layered networks: The tiling algorithm”, Journal of Physics, Vol. A22, pp. 2191-2203, 1989.

[12] S. -U. Guan and S. Li, “An approach to parallel growing and training of neural networks”, in Proceedings of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems, vol. 2, Honolulu, Hawaii, 2000, pp. 1101-1104.

[13] L. Prechelt, “PROBEN1: A set of neural network benchmark problems and benchmarking rules”, Technical Report 21/94, Department of Informatics, University of Karlsruhe, Germany, 1994.

[14] L. Prechelt, “Investigation of the CasCor family of learning algorithms”, Neural Networks, Vol. 10, pp. 885-896, 1997.

[15] M. Riedmiller and H. Braun, “A direct adaptive method for faster backpropagation learning: the RPROP algorithm”, in Proceedings of the IEEE International Conference on Neural Networks, pp. 586-591, 1993.

[16] R. Setiono and H. Liu, “Neural-network feature selector”, IEEE Transactions on Neural Networks, Vol. 8, pp. 654 –662, 1997.

[17] P. V. D. Laar, T. Heskes and S. Gielen, “Partial retraining: a new approach to input relevance determination”, International Journal of Neural Systems, Vol. 9, pp. 75-85, 1999.

[18] S. -U. Guan and S. Li, “An approach to incremental learning with respect to input attributes”, in Proceedings of International ICSC Congress on Intelligent Systems and Applications (ISA'2000), University of Wollongong, Australia, 2000.