View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

ELSEVIER

Artificial Intelligence 75 (1995) 51-62

Artificial Intelligence

Learning of modular structured networks

Masumi Ishikawa

Department of Control Engineering and Science, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology, Iizuka, Fukuoka 820, Japan

Abstract

Learning of large-scale neural networks suffers from computational cost and the local minima problem. One solution to these difficulties is the use of modular structured networks. Proposed here is the learning of modular networks using structural learning with forgetting. It enables the formation of modules. It also enables automatic utilization of appropriate modules from among the previously learned ones. This not only achieves efficient learning, but also makes the resulting network understandable due to its modular character.

In the learning of a Boolean function, the present module acquires information from its subtask module without any supervision. In the parity problem, a previously learned lower-order parity problem is automatically used. The geometrical transformation of figures can be realized by a sequence of elementary transformations. This sequence can also be discovered by the learning of multi-layer modular networks. These examples well demonstrate the effectiveness of modular structured networks constructed by structural learning with forgetting.

1. Introduction

Neural networks are widely used in many fields such as pattern recognition [1,5, 6]. The backpropagation learning (hereafter referred to as BP learning) is the most frequently used learning method in neural networks. It, however, suffers from serious difficulties: computational cost and the local minima problem [4] in case of large-scale homogeneous networks. There have been two alternative approaches to the solution to these difficulties.

The first approach is the use of modular structured networks, i.e., a large-scale network composed of modules, each of which performs a simple task [3]. A module, here, means a group of units which jointly perform a function. It is assumed that only the output of each module is accessible from its outside.

0004-3702/95/$09.50 © 1995 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(94)00061-1

The second approach is the use of innate network structure and connection weights [1]. In this approach, the role of learning is limited to only small perturbations around true values.

In the present paper, I propose a novel method for the learning of modular structured networks using the previously proposed structural learning with forgetting [2]. The essence of the previous proposal is that the forgetting of connection weights makes unnecessary connections fade out, thus generating a skeletal network.

The structural learning with forgetting enables formation of modules by eliminating unnecessary connections due to forgetting. It also enables automatic utilization of appropriate modules from among the previously learned ones. This not only achieves efficient learning, but also makes the resulting network understandable due to its modular character. These characteristics are in sharp contrast to the learning of conventional modular structured networks [5].

In the following chapter, an outline of structural learning with forgetting is presented. Chapter 3 explains the formation and learning of modular structured networks. This is followed by various examples demonstrating the effectiveness of the learning of modular structured networks by structural learning with forgetting: Boolean functions in Chapter 4, the parity problems in Chapter 5 and geometrical transformation of figures in Chapter 6. Chapter 7 concludes the paper.

2. Structural learning with forgetting

A major purpose of structural learning with forgetting is the discovery of regularities and rules in training data. Its essential idea is that forgetting or decay of connection weights enables the emergence of regularities and rules in the form of skeletal network structure. The crucial point is the way how connection weights are decayed at each weight change. The criterion function adopted here is:

$$J_f = \sum_k (o_k - t_k)^2 + \varepsilon' \sum_i \sum_j |w_{ij}| \tag{1}$$

where the first term on the right-hand side is the criterion in the BP learning, i.e., the sum of squared output errors, the second term is a penalty criterion preventing a network from becoming complex, ε' is its relative importance, J_f is a total criterion, w_{ij} is the connection weight from unit j to unit i, o_k is the output of output unit k, and t_k is its target.

The change of the connection weight, w_{ij}, is represented as,

$$\Delta w_{ij} = -\eta \frac{\partial J_f}{\partial w_{ij}} = \Delta w'_{ij} - \varepsilon \,\mathrm{sgn}(w_{ij}) \tag{2}$$

where $\Delta w'_{ij}$ $(= -\eta \partial J/\partial w_{ij})$ is the weight change due to the BP learning, η is a learning rate, ε $(= \eta\varepsilon')$ is the amount of forgetting at each weight change, and $\mathrm{sgn}(x)$ is the sign function, i.e., 1 when x is positive and -1 otherwise.

A quasi-linear form of w_{ij} in Eq. (1) is simple and indicates the preference of simple structured networks over complex ones. Eq. (2) shows that each connection loses its weight by the constant amount, ε, at each weight change. This is the reason why it is named *forgetting*.

Due to the above forgetting, the resulting network is composed of only the connections which are definitely necessary. The resulting skeletal network structure reveals regularity in training data. The learning of networks of various sizes by trial and error is no longer necessary, because *a priori* information on the network structure such as the number of layers and the number of hidden units at each hidden layer is not required.

3. Module formation and learning of modular networks

The learning of modular structured networks here is performed in two stages: the first stage being the learning of connection weights in each module and the second stage being that of inter-module connection weights. Merits of using modular networks are the following. Firstly, because each module is small, computational cost and the local minima problem in the first stage is not serious.

Secondly, computational cost in the second stage is not so serious as that of a homogeneous structured network, because component modules have already been learned.

Thirdly, a resulting modular network is easy to understand, because understanding how component modules are connected to each other is far simpler than understanding how separate units are connected to each other. Whether or not this merit holds depends on learning methods. As shown previously, structural learning with forgetting generates networks with a small number of connections. This property greatly helps to understand the resulting modular networks. The BP learning, however, does not share this property.

Lastly, as will be shown later, learning becomes more efficient as it proceeds, because many previously learned modules become available. This merit is not only useful from a practical point of view, but also interesting from a cognitive science point of view because of its similarity to the learning by humans.

Fig. 1 illustrates how a module is formed. A task is given to a group of units (called output units) from the outside as pairs of input and target outputs. Suppose the input layer contains all the necessary information to do the given task. The network is composed of a group of hidden units in addition to those input and output units, and can freely use these hidden units. The structural learning with forgetting generates a subnetwork with a small number of input and hidden units due to the elimination of connections. The resulting subnetwork in Fig. 1 can be interpreted as a module composed of output units and relevant hidden units.

In contrast to this, the BP learning uses up all the input and hidden units. Therefore the resulting network cannot be interpreted as a module.

It is not a realistic assumption that the learning of a module is based solely on information from the input layer. The more complex a task becomes, the larger the size of a resulting module becomes. This increases the seriousness of computational cost and the local minima problem for the learning of a new module.

A more realistic assumption is that a module, during its learning, may acquire information either from the input layer or from previously learned modules. Fig. 2 illustrates an example of a 2-module network. Suppose module A is to learn a task, and module B

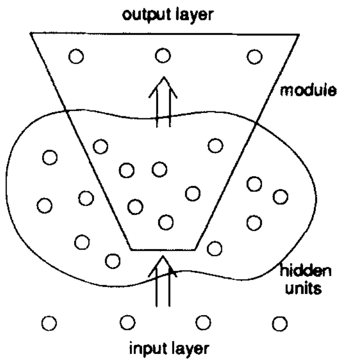

Fig. 1. Formation of a module.

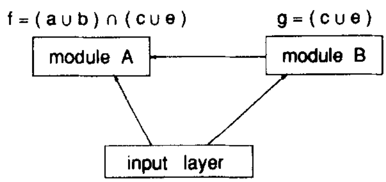

Fig. 2. An example of a network composed of two modules and the input layer.

has already learned its subtask. Suppose further that the input layer contains all the necessary information on those tasks. Module A, during its learning, acquires information on the subtask from module B, not from the input layer. The reason is the following. Information in module B is more condensed than that in the input layer. Therefore a module acquiring information from module B is simpler than the one acquiring information solely from the input layer. The property of structural learning with forgetting enables the generation of the former module.

Generally speaking, if a module uses outputs of previously learned modules, its learning becomes more efficient and a resulting network is constructed by assembling the previously learned modules. The penalty term in Eq. (1) helps the generation of this simplified network.

As learning proceeds, the number of available modules increases. The essential problem is this: from where does a module acquire information for its learning? This problem becomes serious as the number of available modules increases. As will be shown in Section 4.2, appropriate modules can be selected without any supervision, provided structural learning with forgetting is adopted.

4. Modular networks for Boolean functions

The purpose of the learning of Boolean functions is not merely to realize the mapping from input to output, but also to discover forms of Boolean functions based on input and output data.

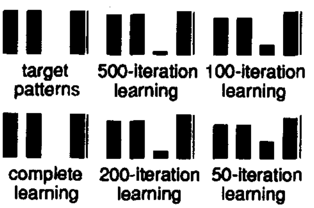

Fig. 3. Output and target output patterns of module B under various degrees of learning performance in terms of the criterion of squared output errors. The height of each bar indicates the output value. Four bars in a row illustrate the output values corresponding to the four training inputs: $(c,e) = (1,0)$, $(0,1)$, $(0,0)$ and $(1,1)$. The length of the solid line on the right signifies the value of 1.

4.1. Sequential learning of a 2-module network

The problem considered here is sequential learning, i.e., one of the modules has finished learning and the other one learns using the outputs of the previously learned module. In other words, the information flow between the two modules is one-directional.

Suppose module A in Fig. 2 is to learn the Boolean function, $f = (a \cup b) \cap (c \cup e)$, where $\cup$ and $\cap$ stand for disjunction and conjunction, respectively. Suppose further that module B has already learned the subtask, $g = (c \cup e)$, to some degree. Without module B, module A acquires information solely from the input layer. The existence of module B is expected to accelerate the learning of module A.

Fig. 3 illustrates output patterns of module B under various degrees of learning performance: complete learning, 500-iteration learning, 200-iteration learning, 100-iteration learning and 50-iteration learning. The output pattern in the complete learning case is the same as the target output pattern. The 50-iteration learning case is quite immature, but can just differentiate between 0 and 1.

In the complete learning case, it is expected that module A uses the output of module B during learning. An interesting question that next arises is whether or not module A acquires information on $(c \cup e)$ from incompletely learned module B instead of that from the input layer. Fig. 4 displays this result: percentages of the connection weight from module B to module A.

Fig. 4 indicates that in all cases except the 50-learning case the connection weights from the input units, c and e, to module A diminishes to zero, and the connection weight from module B to module A increases during learning. In the 50-learning case the learning performance of module B is too poor to be used by module A. In the 100-learning case, although only partly shown in Fig. 4, the connection weight from module B to module A becomes completely dominant after 50,000 iterations. Fig. 4 also illustrates that the better the learning performance of module B is, the faster the connection from module B to module A becomes dominant.

Fig. 5 shows that the higher the learning performance of module B is, the faster the learning of module A becomes. This is in accordance with the difference of the speed at which the connection from module B to module A becomes dominant, as in Fig. 4.

The reorganization of modular structured networks is also noteworthy. Fig. 6 illustrates the result of learning; modules A and B independently learn their tasks using structural

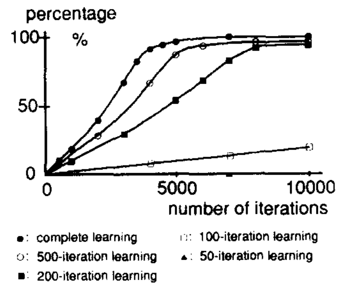

Fig. 4. Percentages of the connection weight from module B to module A, i.e., $100 \times |W_{AB}|/(|W_{AB}| + |W_{A,\mathrm{input}}|)$, where $|W_{AB}|$ is the absolute value of the connection weight from module B to module A, and $|W_{A,\mathrm{input}}|$ is the sum of the absolute values of the connection weights from the input units, c and e, to module A. The vertical axis indicates the percentage. The horizontal axis represents the number of iterations during learning. Since the initial connection weight from module B to module A is set to zero, the initial value of the percentage is also zero. The percentage of connections from the input units, c and e, to module A is calculated by subtracting the above percentage from 100. The parameters of learning are: a learning rate $\eta = 0.1$, a momentum $\alpha = 0.2$ and the amount of forgetting $\varepsilon = 10^{-4}$.

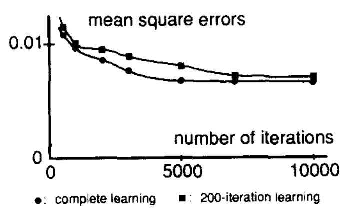

Fig. 5. Learning speed of module A under various degrees of learning performance of module B. The parameters of learning are the same as in Fig. 4.

learning with forgetting based on information from the input layer. Starting from this modular network, structural learning with forgetting generates Fig. 7. These two figures indicate that the initial connections from the input units, c and e, to module A are completely replaced by the connection from module B to module A. This can be regarded as a simple example of the process in which pieces of knowledge are reorganized into more coherent and amalgamated knowledge.

4.2. Sequential learning of a multiple-module network

Suppose a new module may acquire information from either previously learned modules or the input layer. From where does the present module acquire information for its learning?

Module A is to learn the Boolean function, $f = (a \cup b) \cap (c \cup e)$. Five modules, B_1, B_2, B_3, B_4 and B_5, are to learn the Boolean function, $(c \cup e)$, which correspond to the complete-learning, 500-iteration learning, 200-iteration learning, 100-iteration learning and 50-iteration learning, respectively. Fig. 8 displays how the information source on $(c \cup e)$ changes as the learning proceeds. It shows that in the beginning stage information both from the modules, B_i's, and from the inputs, c and e, is used, but as learning proceeds information from the best learned module, B_1, to module A becomes dominant.

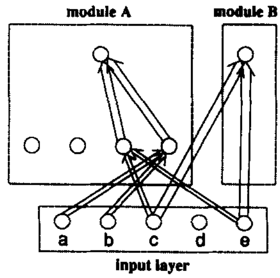

Fig. 6. The independently learned modular structured network. Boldness of each connection is approximately proportional to the absolute value of the corresponding connection weight. Module A learns $f = (a \cup b) \cap (c \cup e)$, and module B learns $g = (c \cup e)$. The parameters of learning are the same as in Fig. 4.

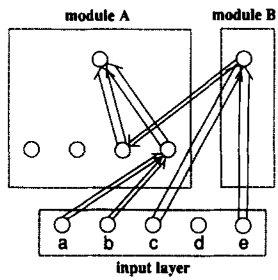

Fig. 7. The modular structured network after reorganization. Module B learns $g = (c \cup e)$, and module A learns $f = (a \cup b) \cap (c \cup e)$ using the output of module B. The parameters of learning are the same as in Fig. 4.

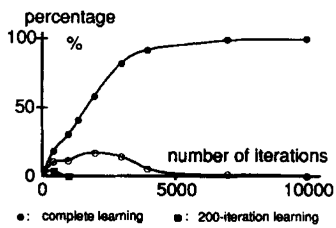

Fig. 8. The percentage of the connection weight concerning $(c \cup e)$ from each module, B_i, to module A. The cases where the percentage is almost zero throughout learning are not shown here. The initial connection weight from each module, B_i, to module A is set to zero. The percentage of connection weights from the inputs, c and e, to module A is calculated by subtracting the sum of the above percentages from 100. The parameters of learning are the same as in Fig. 4.

In other words, in cases where multiple modules are available, the best module for the learning of the present module is automatically selected, provided structural learning with forgetting is used. This property of the appropriate selection without any supervision is desirable for the learning of modular structured networks.

In case of the BP learning, on the other hand, the information on $(c \cup e)$ comes from all the modules and the input layer, as in Fig. 9. Generally, in the BP learning the

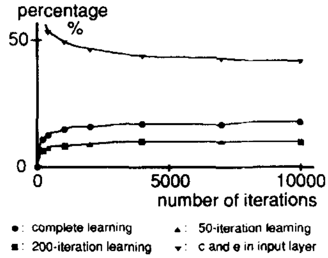

Fig. 9. The percentage of the connection weight concerning $(c \cup e)$ from each module, B_i, to module A. Not all the cases are shown here for clarity. The parameters of learning are: the learning rate $\eta = 0.1$ and the momentum $\alpha = 0.2$.

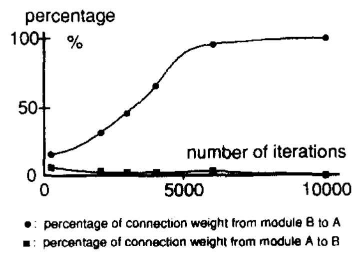

Fig. 10. The percentage of the connection weight concerning $(c \cup e)$ from one module to the other. The parameters of learning are the same as in Fig. 4.

present module acquires information from modules whose outputs have correlation with the present target output. For this reason, the BP learning is not suited for the learning of modular networks.

4.3. Concurrent learning of a 2-module network

Here two modules learn concurrently, i.e., each module can use the output of the other one. Suppose module A is to learn the Boolean function, $f = (a \cup b) \cap (c \cup e)$, and module B is to learn $g = (c \cup e)$. These two modules start learning at the same time. Qualitatively speaking, the learning of module B precedes that of module A due to its simplicity.

In the beginning stage of learning, both modules acquire information on $(c \cup e)$ from the input layer. Since the learning of module B becomes almost complete in about 2,000 iterations, module A begins to acquire information on $(c \cup e)$ from module B instead of acquiring it from the input layer. Fig. 10 illustrates that the percentage of the connection weight from module B to module A increases and that from module A to module B diminishes to zero as learning proceeds. The percentage of connection weights from the input layer to module A is calculated by subtracting the above percentage from 100.

This result shows that even if multiple modules learn concurrently, a simple modular network can emerge, provided there exist part-whole relations among tasks to be learned.

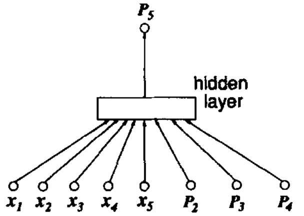

Fig. 11. Learning of the parity problem of order 5 using the modular structured network. The parameters of learning are the same as in Fig. 4.

5. Modular networks for parity problems

A parity problem is to judge whether the number of 1's in a binary input vector is odd or even; when it is odd, the output is 1 and when it is even, the output is 0. The parity problem with n inputs is called the parity problem of order n.

As the order of a parity problem increases, its learning becomes more difficult due to the local minima problem. When a parity problem of some order has already been solved, it helps solve parity problems of higher orders because of the similarity among parity problems of various orders.

In this chapter, a trial is performed to solve a high-order parity problem under the assumption that lower-order parity problems have already been solved and their outputs are available for its learning.

Let the parity problems of orders 2, 3 and 4 be P_2, P_3 and P_4, respectively, and their inputs be (x_1, x_2), (x_1, x_2, x_3) and (x_1, x_2, x_3, x_4), respectively. Suppose, in the learning of the parity problem of order 5, P_5, not only the inputs, $(x_1, x_2, x_3, x_4, x_5)$, but also the outputs of lower-order parity problems, P_2, P_3 and P_4 are available as shown in Fig. 11. The structural learning with forgetting using all 32 $(= 2^5)$ training samples generates the modular network constructed as the *exclusive or* of two inputs: P_4, which has the largest similarity to P_5 among modules and the additional input unit, x_5. It clearly reveals the structure of the parity problem P_5. It is to be noted that the inputs, (x_1, x_2, x_3, x_4), and lower-order parity problems, P_2 and P_3, are not used in solving the parity problem, P_5.

This result can be extended to parity problems of higher orders. The parity problem of order n, P_n, is represented as the modular network constructed as the *exclusive or* of P_{n-1} and the additional input unit, x_n.

6. Modular networks for geometrical transformation of figures

The property that only the appropriate modules among previously learned ones are used and all the connections from other modules diminish without any supervision is useful in the learning of modular structured networks. Up to now all the previously learned modules are located at the same level as the input layer. In other words, only as it were flat modular networks have been considered.

In this chapter, the learning of a sequence of modules by dynamically concatenating

Table 1
Modules for elementary geometrical transformations of figures.

Module	Elementary transformation
M_1	no transformation
M_2	rotation of 90° clockwise
M_3	rotation of 180° clockwise
M_4	rotation of 270° clockwise
M_5	rightward translation by one block
M_6	leftward translation by one block
M_7	upward translation by one block
M_8	downward translation by one block

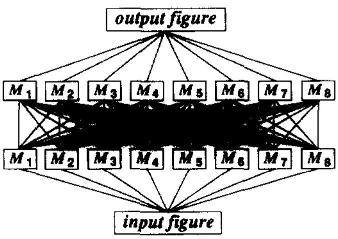

Fig. 12. A 2-layer modular structured network. Each module layer is composed of eight modules. Each module is realized by a 2-layer subnetwork.

previously learned modules is considered. As an example, geometrical transformation of simple figures is adopted here. Table 1 shows a list of elementary geometrical transformations of figures. Each of these geometrical transformations corresponds to a module.

Here, a simple geometrical transformation realized by concatenating two modules is considered. Fig. 12 illustrates an example of a multi-layer modular structured network. Both the first and the second module layers have 8 modules in Table 1. Each module can easily be realized by a 2-layer neural subnetwork and is held fixed during the learning of the modular network. At first glance Fig. 12 seems to be a 6-layer network, but since the input layer and the lower layer of the first module layer are one and the same layer, it actually is a 5-layer network. In Fig. 12 all the layers except the output layer use linear units instead of sigmoidal units.

The problem considered here is to discover a sequence of geometrical transformation modules based on four pairs of input and output figures in Fig. 13. The supposed solution to this problem is the rotation of 90° clockwise followed by the upward translation by one block.

If the BP learning is used in this problem, the realization of the mapping from input figures to output figures is straightforward, but the resulting inter-module connections cannot be interpreted as a sequence of geometrical transformations. Instead, they correspond to a complex combination of all the transformations and are impossible to understand. Therefore structural learning with forgetting plays an essential role in discovering a sequence of geometrical transformation modules.

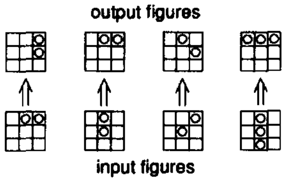

Fig. 13. Input figures and target output figures. The former is the figure before transformation and the latter is the one after transformation.

Given four pairs of input figures and target output figures, inter-module connections are trained. The structural learning with forgetting generates a single path or at most a few parallel paths from input to output, explicitly presenting a sequence of geometrical transformation modules.

Ten simulations with different initial connection weights are tried. The solution of leftward translation by one block followed by rotation of 90° clockwise is obtained five times. In this case, only the connections from M_6 in the first module layer to M_2 in the second module layer and the connections from this M_2 to the output layer remain, and all other connections fade out. In three out of ten cases, the supposed solution of the rotation of 90° clockwise followed by the upward translation by one block is obtained. In two out of ten cases, both solutions appear simultaneously.

Generally speaking, it is impossible to know *a priori* how many layers of modules are necessary. Preparing an excess number of module layers solves this problem; if the number of module layers is too large, no transformation, M_1, in Table 1 can be used for necessary number of times.

In cases where a given task does not depend on a *sequence* of modules, the problem is reduced to the discovery of a subset of modules instead of a sequence of modules. In other words, only one module layer is sufficient to solve this kind of problems.

7. Conclusions

In the paper, I have demonstrated that structural learning with forgetting can efficiently construct modular networks in various examples: Boolean functions, parity problems, and the transformation of geometrical figures. The property that appropriate modules can automatically be selected without any supervision is especially important in the learning of modular networks.

The use of previously learned modules helps shorten the required learning time. It also simplifies the resulting modular networks, because previously learned modules can be utilized as building blocks. This could be seen as a model of progressive learning based on the previously learned results and is similar to the *layers of society* in the society of mind [3].

References

[1] K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern

recognition unaffected by shift in position, *Biol. Cybern.* **36** (1980) 193-202.
[2] M. Ishikawa, A structural learning algorithm with forgetting of link weights, *IJCNN* (Washington DC, 1989), II-626; also: *Tech. Report* TR-90-7, Electrotechnical Laboratory, Japan (1990).
[3] M. Minsky, *The Society of Mind* (Simon and Schuster, New York, 1985).
[4] M. Minsky and S. Papert, *Perceptrons* (MIT Press, Cambridge, MA, 1988).
[5] Y. Mori and K. Joe, A large-scale neural network which recognizes handwritten Kanji characters, in: D.S. Touretzky, ed., *Advances in Neural Information Processing Systems* **2** (1990) 415-422.
[6] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, eds., *Parallel Distributed Processing*, Vols. 1 and 2 (MIT Press, Cambridge, MA, 1986).