Emergence of Learning: An Approach to Coping with NP-Complete Problems in Learning

Bao-Liang Lu and Michinori Ichikawa
Lab. for Brain-Operative Device, RIKEN Brain Science Institute
2-1 Hirosawa, Wako-shi, 351-0198, Japan
{lu; ichikawa}@brainway.riken.go.jp

Abstract: Various theoretical results show that learning in conventional feedforward neural networks such as multilayer perceptrons is NP-complete. In this paper we show that learning in min-max modular (M^3) neural networks is tractable. The key to coping with NP-complete problems in M^3 networks is to decompose a large-scale problem into a number of manageable, independent subproblems and to make the learning of a large-scale problem emerge from the learning of a number of related small subproblems.

1 Introduction

One of the most important issues in supervised learning for feedforward neural networks is the computational complexity of the training problem, which asks how much computational effort is required to achieve good performance in the training phase. Judd [2, 3] showed that the following *loading problem* to be NP-complete: given a network specification and a set of training examples, does there exist a set of adjustable parameters for the network so that the network produces the correct output for all the examples ? Under the framework of Judd's loading problem, various theoretical results have been reported. All the results suggest that learning in feedforward neural networks is intractable [1, 4, 6].

On the other hand, using a refinement of the Probably Approximately Correct (PAC) learning model, Maass [8] proposed multilayer neural networks with piecewise polynomial activations and a fixed number of analog inputs, and showed that efficient learning in these networks is possible. However, it seems that how to select suitable single homogeneous neural networks for large-scale problems is still problematic.

In fact, neuroanatomy and cognitive neuroscience provide a great amount of evidence showing that the information processing system in the brain consists of modules, which can function quite independently of each other. Modularity appears to be an important principle in the architecture of the brain [11], and a key to understanding the emergence of learning in artificial neural networks [9].

This paper addresses the computational complexity of supervised learning problem for min-max modular (M^3) neural networks. The M^3 network is a hierarchical, parallel, and modular learning framework proposed in our previous work [7]. Fig. 1 presents an overview of the M^3 learning framework. In this paper we show that learning in M^3 networks is tractable. The key to coping with NP-complete problems in learning of M^3 networks is to decompose a large-scale problem into a number of manageable subproblems and to make the learning of large-scale problem emerge from the learning of a number of related small subproblems.

2 Task decomposition

Let T be the training set for a K-class classification problem,

$$T = \{(\boldsymbol{x}_t, \boldsymbol{y}_t)\}_{t=1}^{N}, \tag{1}$$

where $\boldsymbol{x}_t \in \mathbf{R}^n$ is the input vector, $\boldsymbol{y}_l \in \mathrm{R}^K$ is the desired output, and N is the total number of training data.

We suggest that a K-class problem as defined in (1) can be divided into $\binom{K}{2}$ relatively smaller two-class subproblems [7]. The training set for each of the subproblems is given by

$$T_{ij} = \{(\boldsymbol{x}_l^{(i)}, 1-\epsilon)\}_{l=1}^{L_i} \cup \{(\boldsymbol{x}_l^{(j)}, \epsilon)\}_{l=1}^{L_j} \text{ for } i = 1, \cdots, K \text{ and } j = i+1, \cdots, K \tag{2}$$

0-7695-0619-4/00 $10.00 © 2000 IEEE

where ϵ is a small real positive number, $\boldsymbol{x}_l^{(i)} \in \mathcal{X}_i$ and $\boldsymbol{x}_l^{(j)} \in \mathcal{X}_j$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively, and L_i denotes the number of data in $\mathcal{X}_i$ for $i = 1, \cdots, K$. If the desired output of a training data is $1-\epsilon$, then the training data is called *positive* training data. Otherwise, *negative* training data.

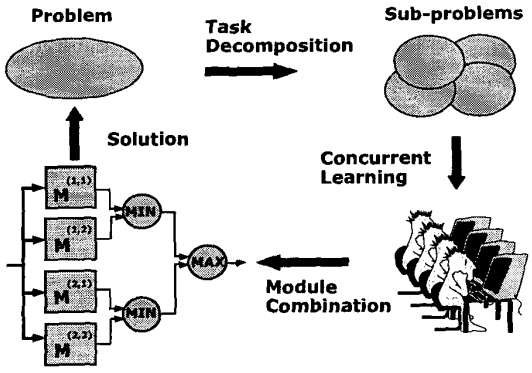

Figure 1: An overview of the min-max modular learning framework.

Assume that the input set $\mathcal{X}_i$ is partitioned into N_i $(1 \leq N_i \leq L_i)$ subsets in the form

$$\mathcal{X}_{ij} = \{\boldsymbol{x}_l^{(ij)}\}_{l=1}^{L_i^{(j)}} \quad \text{for } j = 1, \cdots, N_i \text{ and } i = 1, \cdots, K, \tag{3}$$

where $\boldsymbol{x}_l^{(ij)} \in \mathbf{R}^n$ is the training input and $\cup_{j=1}^{N_i} \mathcal{X}_{ij} = \mathcal{X}_i$.

Various methods can be used for partitioning $\mathcal{X}_i$ into N_i subsets [7]. A simple and straightforward approach is to divide $\mathcal{X}_i$ randomly. In this case, no domain specialists or a prior knowledge concerning the decomposition of the problems are required. We use the random decomposition method throughout this paper.

According to the above partition of $\mathcal{X}_i$, a K-class classification problem can be divided into

$$\sum_{i=1}^{K} \sum_{j=i+1}^{K} N_i \times N_j \tag{4}$$

relatively smaller and simpler two-class subproblems. The training set for each of the subproblems is given by

$$\begin{aligned} T_{ij}^{(u,v)} = &\{(\boldsymbol{x}_l^{(iu)}, 1-\epsilon)\}_{l=1}^{L_i^{(u)}} \cup \{(\boldsymbol{x}_l^{(jv)}, \epsilon)\}_{l=1}^{L_j^{(v)}} \\ &\text{for } u = 1, \cdots, N_i,\ v = 1, \cdots, N_j,\ i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K \end{aligned} \tag{5}$$

where $\boldsymbol{x}_l^{(iu)} \in \mathcal{X}_{iu}$ and $\boldsymbol{x}_l^{(jv)} \in \mathcal{X}_{jv}$ are the training inputs belonging to class $\mathcal{C}_i$ and class $\mathcal{C}_j$, respectively.

If the training set $T_{ij}^{(u,v)}$ has only two different elements in the form

$$\begin{aligned} T_{ij}^{(u,v)} = &\{(\boldsymbol{x}_1^{(iu)}, 1-\epsilon) \cup (\boldsymbol{x}_1^{(jv)}, \epsilon)\} \\ &\text{for } u = 1, \cdots, L_i,\ v = 1, \cdots, L_j,\ i = 1, \cdots, K, \text{ and } j = i+1, \cdots, K, \end{aligned} \tag{6}$$

then this training set is obviously a linearly separable problem because any two different training data can always be separated by a hyper-plane.

An important feature of the above task decomposition method is that each of the subproblems can be treated as a completely independent, non-communicating subproblem in the training phase. Consequently, all of the subproblems can be learned in a completely concurrent way.

3 Module combination

After training of each of the network modules on the related subproblems, all of the individual trained modules can be easily integrated into an M^3 network according to the following module combination principles.

Theorem 1 (Minimization Principle) *Suppose a two-class problem B is divided into P relatively smaller two-class subproblems, B_i for $i = 1, \cdots, P$, and also suppose that all the subproblems have the same positive training data and hold different negative training data. If the P subproblems are correctly learned by the corresponding P individual network modules, M_i for $i = 1, \cdots, P$, then the combination of the P trained network modules with an* **MIN** *unit produces the correct output for all the training inputs in B, where the function of the* **MIN** *unit is to find a minimum value from its multiple inputs.*

Proof. Let us consider the problem as the following two cases.

Case 1: Let $\boldsymbol{x}$ be a positive training input belonging to B. According to the definition of B_i, all the P subproblems, B_i for $i = 1, \cdots, P$, contain the *same* $\boldsymbol{x}$. Since all of the subproblems are successfully learned by the corresponding P networks modules, M_i for $i = 1, \cdots, P$, the output of each of the P network modules, $h_i(\boldsymbol{x})$ for $i = 1, \cdots, P$, satisfies

$$| h_i(\boldsymbol{x}) - (1 - \epsilon) | \leq \delta \quad \text{for } i = 1, \cdots, P, \tag{7}$$

where δ is a positive real number, which denotes the error tolerance.

Therefore, we have

$$| \min_{i=1}^{P} h_i(\boldsymbol{x}) - (1 - \epsilon) | \leq \delta \tag{8}$$

Case 2: Let $\boldsymbol{x}$ be a negative training input belonging to B. According to the definition of B_i, among the P subproblems, there are at most Q ($1 \leq Q < P$) subproblems, $B_{q(i)}$ for $i = 1, \cdots, Q$, contain $\boldsymbol{x}$, and the other R ($R = P - Q$) subproblems, $B_{r(i)}$ for $i = 1, \cdots, R$, do not hold $\boldsymbol{x}$. Since all of the subproblems are successfully learned by the corresponding networks modules, the outputs of Q modules satisfy

$$| h_{qi}(\boldsymbol{x}) - \epsilon) | \leq \delta \quad \text{for } i = 1, \cdots, Q. \tag{9}$$

On the other hand, because $\boldsymbol{x}$ was not used as a training data for the other R network modules in the training phase, each of the R trained network modules might produce *arbitrary* output for $\boldsymbol{x}$, that is,

$$0 \leq h_{ri}(\boldsymbol{x}) \leq 1 \quad \text{for } i = 1, \cdots, R, \tag{10}$$

where the assumption we made is that the range of the output for each of the network modules is between 0 and 1.

From (9) and (10), we have

$$| \min_{i=1}^{P} h_i(\boldsymbol{x}) - \epsilon) | \leq \delta \tag{11}$$

□

Theorem 2 (Maximization Principle) *Suppose a two-class problem B is divided into P relatively smaller two-class subproblems, B_i for $i = 1, \cdots, P$, and also suppose that all the subproblems have the same negative training data and hold different positive training data. If the P subproblems are correctly learned by the corresponding P individual network modules, M_i for $i = 1, \cdots, P$, then the combination of the P trained network modules with an* **MAX** *unit produces the correct output for all the training input in B, where the function of the* **MAX** *unit is to find a maximum value from its multiple inputs.*

Proof. The proof of this Theorem is omitted due to space requirements. It can be proved following the similar way as mentioned in Theorem 1. □

Let $\boldsymbol{y}$ denote the actual output vector of the M^3 network for a K-class classification problem, and let $g(\boldsymbol{x})$ denote the transfer function of the M^3 network. We may then write

$$\boldsymbol{y} = g(\boldsymbol{x}) = [g_1(\boldsymbol{x}), \cdots, g_K(\boldsymbol{x})]^T, \tag{12}$$

where $\boldsymbol{y} \in \mathbf{R}^K$, and $g_i(\boldsymbol{x}) \in \mathbf{R}$ is called the *discriminant function*, which discriminates the patterns of class C_i from those of the rest classes.

Table 1: Performance comparison of LeNet and the M^3 network on the handwritten digit problem. Note that the CPU time of LeNet was measured on SUN SPARCstation 1 [5], while the CPU time of the M^3 network was measured on SUN Ultra 30.

Classifiers	Error rates (%)		CPU time (sec.)	
	Training	Test	Max.	Total
LeNet	1.1	4.3	259200	259200
M^3	0.0	5.0	48	9401

According to the minimization principle, the discriminant functions $g_i(\boldsymbol{x})$ of the M^3 network for the $\binom{K}{2}$ two-class subproblems defined in (2) can be given by

$$g_i(x) = \min\left[\min_{j=i+1}^{K} h_{ij}(x), \min_{r=1}^{i-1}(b - h_{ri}(x))\right], \tag{13}$$

where the term $b - h_{ri}(x)$ denotes the inverse of $h_{ri}(x)$ and b denotes the upper limit of the output value of each module.

Similarly, according to the minimization and maximization principles, the discriminant function $g_i(\boldsymbol{x})$ of the M^3 network for $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems defined in (5) can be expressed as

$$g_i(x) = \min\left[\min_{j=i+1}^{K}\left[\max_{k=1}^{N_i}\left[\min_{l=1}^{N_j} h_{ij}^{(k,l)}(x)\right]\right], \min_{r=1}^{i-1}\left(b - \max_{k=1}^{N_r}\left[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(x)\right]\right)\right], \tag{14}$$

where the term $b - \max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(x)]$ denotes the inverse of $\max_{k=1}^{N_r}[\min_{l=1}^{N_i} h_{ri}^{(k,l)}(x)]$.

4 Coping with NP-Complete Problems

From experience of training neural networks, we know that small problems can be easily learned by means of existing learning algorithms. For example, it is easy for us to learn the XOR problem by using a three-layer perceptron and the back-propagation algorithm. Therefore, we naturally ask: can the learning of a large-scale problem emerge from the learning of a number of corresponding small subproblems ? We will give a positive answer to this question from the following two theorems.

Theorem 3 (Arbitrary Partition) *Suppose that there is a learning algorithm $\mathcal{A}_\mu$ which is efficient for learning two-class problems whose sizes are equal to or less than μ. If a K-class problem T has a finite number (N) of training data, and if T is divided into a number of two-class subproblems according to (2) and (5), then for μ ($2 \le \mu \le N$), there exist K integers, N_i ($1 \le N_i \le L_i$) for $i = 1, \cdots, K$, that make the size of each of $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems be equal to or less than μ.*

Proof. Let $\alpha = \max\{\lceil L_1/N_1\rceil, \cdots, \lceil L_K/N_K\rceil\}$, where $\lceil z\rceil$ denotes the smallest integer greater than or equal to z. If $\alpha \le \mu/2$, then the number of training data in $T_{ij}^{(u,v)}$ of (5) is equal to or less than μ. Therefore, let $N_i = \lfloor 2L_i/\mu\rfloor$, then the size of each of the subproblems is equal to or less than μ. □

In practical applications of neural networks, the value of μ might depend on several factors such as the probability distribution of training data, the capability of network modules, the training algorithms, and the computer power available.

Theorem 4 (Guaranteed Integration) *If a K-class problem T is divided into $\sum_{i=1}^{K}\sum_{j=i+1}^{K} N_i \times N_j$ two-class subproblems according to (2) and (5), and each of the subproblems is correctly learned by a corresponding network module, then there is an M^3 network which is just a combination of all of the network modules, $\sum_{i=1}^{K}\sum_{r=1}^{i-1} N_i \times N_r$* **INV** *units, $(K + \sum_{i=1}^{K} N_i)$* **MIN** *units, and $\binom{K}{2}$* **MAX** *units, such that the M^3 network produces the correct output for all the training inputs in T.*

Proof. The results can be proved by using the minimization and maximization principles. □

5 Experimental Results

5.1 XOR Problem

According to the task decomposition method, the XOR problem (see Fig. 2(a)) was divided into four linearly separable subproblems: $T^{(1,1)}$, $T^{(1,2)}$, $T^{(2,1)}$, and $T^{(2,2)}$, which are depicted in Fig. 2(b)-(e), respectively. Four perceptrons represented as $M^{(1,1)}$, $M^{(1,2)}$, $M^{(2,1)}$, and $M^{(2,2)}$ were selected to learn $T^{(1,1)}$, $T^{(1,2)}$, $T^{(2,1)}$, and $T^{(2,2)}$, respectively. The M^3 network which consists of the four perceptrons is shown in Fig. 1. The *optimal boundaries* formed by the four perceptrons are shown in Fig. 3(a)-(d), respectively. The responses of the combinations of individual modules and the whole M^3 network are shown in Fig. 3(e)-(g), respectively. Comparing Fig. 2(a) with Fig. 3(g), we see that the M^3 network forms *optimal boundaries* for the XOR problem.

Figure 2: Partition of the XOR problem into four linearly separable subproblems. (a) The training inputs for the original XOR problem, (b) $T^{(1,1)}$, (c) $T^{(1,2)}$, (d) $T^{(2,1)}$, and (e) $T^{(2,2)}$, respectively. The black and white points represent the inputs whose desired outputs are '0' and '1', respectively, and the grey represents only the background of the figures.

Figure 3: The responses of (a) $M^{(1,1)}$, (b) $M^{(1,2)}$, (c) $M^{(2,1)}$, (d) $M^{(2,2)}$, (e) the combination of $M^{(1,1)}$ and $M^{(1,2)}$ with the MIN unit, (f) the combination of $M^{(2,1)}$ and $M^{(2,2)}$ with the MIN unit, and (g) the whole M^3 network, respectively. The black and white represent the outputs of '0' and '1', respectively.

5.2 Handwritten Digit Recognition Problem

The training set and test set for the handwritten digit recognition problem consist of 7291 and 2007 data, respectively. The image for each handwritten ZIP code data contains 16 pixel rows by 16 pixel columns, for a total 256 pixels.

In [5], LeCun, *et al.*, reported that three days were required for training a specific five-layer feedforward neural network (LeNet) on this problem[1]. Here, the original problem is decomposed into 9514 subproblems randomly, where $N_1 = 24$, $N_2 = 20$, $N_3 = 15$, $N_4 = N_5 = N_7 = N_8 = N_{10} = 13$, and $N_6 = N_9 = 11$. The number of training data in each of the subproblems is about 100. In the simulation, 9514 three-layer MLPs were selected for learning the corresponding subproblems. Each of the MLPs has five hidden units. All of the MLPs were trained by the backpropagation algorithm [10]. The numbers of iterations and CPU times (sec.) required for training the 9514 modules are shown in Figs. 4(a) and 4(b), respectively. From Fig. 4(b), we see that each of 7372 subproblems can be learned within two seconds. The maximum CPU time (see Table 1) for learning a single subproblem is about 48 seconds. This means that to solve the handwritten digit recognition problem requires only 48 seconds, instead of three days, if all of the subproblems are learned in a complete parallel way. The total CPU time used for learning all 9514 subproblems and the performance of the M^3 network are shown in Table 1.

[1] In [5], 7291 handwritten digits and 2549 printed digits were used as training data, while only 7291 handwritten digits were used here.

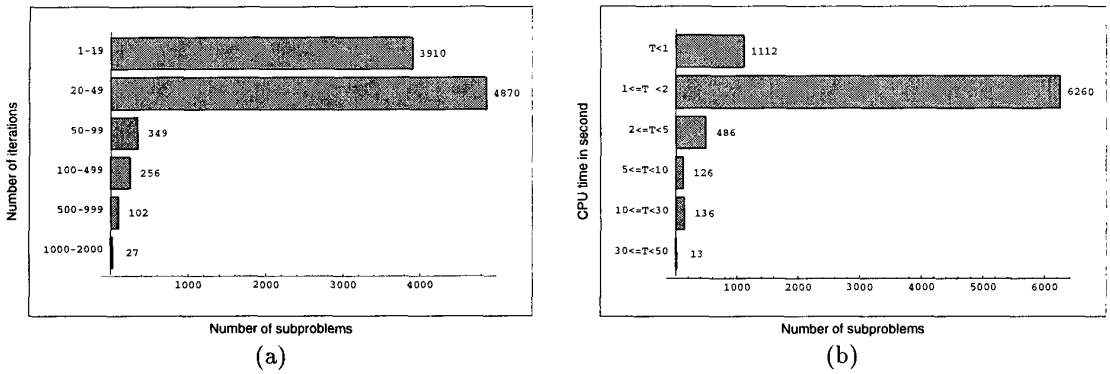

Figure 4: Convergence of the 9514 network modules for learning the corresponding subproblems. (a) shows that most of the subproblems were successfully learned within 50 iterations. (b) shows that most of the subproblems were correctly learned within two seconds.

6 Conclusions

By breaking through the non-modular learning architecture of Judd's loading problem, we have presented a more powerful modular learning framework, namely M^3 learning framework. We show in this paper that the learning of a large-scale problem in M^3 neural networks can emerge from the learning a number of related small subproblems. Since any large-scale problems can be easily divided into a number of independent subproblems as small as we expect and all of the subproblems can be learned in a completely concurrent way, learning in M^3 neural networks is easy ! The importance of the result lies in the facts that it provides us with a new approach to coping with NP-complete problems in neural network learning and it gives us an example showing the mechanism of emergence of learning in neural networks.

References

[1] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is NP-complete", *Neural Networks*, vol. 5, pp. 117-127, 1992.

[2] J. S. Judd, "Learning in networks is hard", *Proc. of 1st International Conference on Neural Networks*, pp. 685-692, IEEE, San Diego, California, June 1987.

[3] J. S. Judd, *Neural Network Design and the Complexity of Learning*, MIT Press, 1990.

[4] D. R. Hush, "Training a sigmoidal node is hard", *Neural Computation*, vol. 11, pp. 1249-1260, 1999.

[5] Y. LeCun et al., "Handwritten digit recognition with a back-propagation network", in *Neural Information Processing Systems*, D. S. Touretzky, Ed. San Mateo, CA: Morgan Kaufmann, pp. 396-404, 1990.

[6] J. H. Lin and J. S. Vitter, "Complexity results on learning by neural nets", *Machine Learning*, vol. 6, pp. 211-230, 1991.

[7] B. L. Lu and M. Ito, "Task decomposition and module combination based on class relations: a modular neural network for pattern classification", *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1244-1256, 1999.

[8] W. Maass, "Agnostic PAC-learning of function on analog neural networks", *Neural Computation*, vol. 7, pp. 1054-1078, 1995.

[9] M. Minsky, *The Society of Mind*, New York: Simon and Schuster, 1986.

[10] D. E. Rumelhart, G. E. hinton, and R. J. Williams, "Learning internal representation by error propagation", in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1, D. E. Rumelhart, J. L. McClelland and PDP Researh Group Eds., Cambridge, MA: MIT Press, pp. 318-362, 1986.

[11] D. C. Van Essen, C. H. Anderson, and D. J. Felleman, "Information processing in the primate visual systems: an integrated systems perspective", *Science*, vol. 255, pp. 419-423, 1992.