A novel Bayesian learning method for information aggregation in modular neural networks

Pan Wang\textsuperscript{a,b}, Lida Xu\textsuperscript{a,c,d,e}, Shang-Ming Zhou\textsuperscript{e}, Zhun Fan\textsuperscript{f}, Youfeng Li\textsuperscript{b}, Shan Feng\textsuperscript{g}

\textsuperscript{a}Institute of Systems Science and Engineering, Wuhan University of Technology, Wuhan 430070, China
\textsuperscript{b}School of Automation, Wuhan University of Technology, Wuhan 430070, China
\textsuperscript{c}School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China
\textsuperscript{d}Department of Information Technology and Decision Sciences, Old Dominion University, Norfolk, VA 23529, USA
\textsuperscript{e}UKCRC DECIPHer (Development and Evaluation of Complex Interventions for Public Health Improvement) Centre, Health Information Research Unit (HIRU), School of Medicine, University of Wales Swansea, SA2 8PP, UK
\textsuperscript{f}Department of Management Engineering, Technical University of Denmark, Lyngby, Denmark
\textsuperscript{g}Institute of Systems Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

\section*{A R T I C L E  I N F O}

Keywords:
Bayesian learning
Modular neural network
Information aggregation
Combination
Modularity

\section*{A B S T R A C T}

Modular neural network is a popular neural network model which has many successful applications. In this paper, a sequential Bayesian learning (SBL) is proposed for modular neural networks aiming at efficiently aggregating the outputs of members of the ensemble. The experimental results on eight benchmark problems have demonstrated that the proposed method can perform information aggregation efficiently in data modeling.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

A modular neural network (MNN) is a neural network model that is characterized by a group of independent sub-neural networks moderated by some intermediaries in the entire architecture to perform complex functions (Jordan & Jacobs, 1991a, 1991b). Each independent sub-network serves as a module and operates on separate inputs to accomplish some subtasks of the task that the network is to perform. An additional layer acting as an intermediary takes the outputs of each module and processes them to produce the output of the network as a whole. In contrast to the “global” neural network models such as back-propagation neural networks (Haykin, 1999) and recurrent neural networks (Zhou & Xu, 1999, 2001), an MNN offers some significant advantages (Feng & Wang, 2007; Wang, Feng, & Fan, 2008). First it has strong biological backgrounds. Cognitive, neuropsychological and neurobiological sciences have indicated that biological brain function exhibits the property of modularity. Some neuropsychological experiments show that a circumscribed brain lesion could cause a specific disorder of language while other brain cognitive functions remain intact (Eccles, 1984; Edelman, 1979). The brain also performs the function of dividing the complex task of visual perception into many subtasks (Happel & Murre, 1994). Interestingly, the neuropsychology indicates that the thalamus of the brain is divided into different layers that separately process color and contrast (Hubel & Livingstone, 1990). Hence, according to the cognitive, neuropsychological and neurobiological sciences, the regions of animal and human brains are organized into specially and functionally segregated modules, while the MNN offers a computational intelligence technique for emulating these functions of segmentation and modularization found in the brain. The second advantage of MNN is that as a computing scheme, it provides the capability of dividing a large, unwieldy neural network into smaller, more manageable components. This is very useful in neural network applications since many real-world problems appear intractable for practical purposes by a single neural network as its size increases. If a complex task is further separated into distinct parts, the possible connections each node can make in building up a network are limited, so the sub-tasks will perform more efficiently than trying to tackle the whole task at once by a single network. As a matter of fact, even though a large neural network modelled by a large number of parameters can be constructed to tackle a complex problem, the network training process could suffer from interference when new data turns up, as these new data can dramatically alter existing connections or just serve to confuse. On the hand, if a task is divided into subtasks to be solved by independent sub-neural networks, each sub-neural network can be tailored for its task with its unique training data. In this way, more effective computing performance can be achieved. Currently, MNN has many applications in the domains such as pattern recognition and classification (Drake & Packianather, 1998; Perez & Galdames, 1998; Zhou, Wu, & Tang, 2009).
image processing (Van Hulle & Tollenacré, 1993; Wang, Der, & Nasrabadi, 1998), system identification (Srinivas, Jeffrey, Huang, Phillips, & Wechsler, 2000), language/speech processing (Chen & Chi, 1999; Sibte & Abidi, 1996), control (Deepak, 1999), prediction/modelling (Pan & Sensen, 2005; Wang, Fan, & Li, 2006), target detection/recognition (Wang et al., 1998), fault diagnosis (Kim & Park, 1993), etc.

In MNN applications, after the ensemble of neural networks are trained, an important issue is how to effectively aggregate the individual network outputs. In fact, knowledge and information aggregation has become a subject of intensive research due to its practical and academic significance in many domains such as decision making (Carvalho & Costa, 2007; Li, 1999a, 1999b; Li & Li, 1999; Xu, 1988; Xu, Liu, & Jones, 2008; Zhao & Li, 2009; Zhou, Chiclana, John, & Garibaldi, 2008), information fusion (Tahani & Salamon, 1990; Hashem, 1997; Perrone & Cooper, 1994; Xu, Krzyzak, & Suen, 1992). Among them the simple averaging (Hashem, 1997) for aggregating the outputs of the members of the ensemble are commonly used. However, few efforts have been made to address this issue from the perspective of Bayesian reasoning, although some researchers have proposed Bayesian schemes for designing “global” neural networks (Buntine & Weigend, 1991a, 1991b) if the computation performed by the network can be decomposed into two or more modules (subsystems) that operate on distinct inputs without communicating with each other. The outputs of the modules are mediated by an integrating unit that is not permitted to feed information back to the modules. In particular, the integrating unit determines (1) how the modules are combined to form the final output of the system, and (2) which modules should learn which training patterns. The corresponding network architecture is illustrated in Fig. 1, in which \(x \in D \subseteq \mathbb{R}^{n \times 1}\) is the input vector; \(y\) is the output, \(\{\text{Net}_i\}_{i=1}^{K}\) represent the individual sub-nets, and \(\{w_i\}_{i=1}^{K}\) are the corresponding combination weights for aggregating the outputs of individual sub-nets to obtain the overall outputs.

\[
y = \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=1}^{K} w_i}
\]  

where \(w_i \geq 0\). These weights could be viewed as the measure of “goodness” of the sub network’s behavior in the entire system. A larger combination weight indicates the associated ensemble member playing a more important role in the decision making process.

In the following, based on sequential Bayesian decision analysis a Bayesian learning approach to MNN is proposed to combine the output(s) of the members of the ensemble of a modular neural network.

First, if the unknown vector has prior density, the posterior density \(p(\theta|x)\) determined by a certain observed vector \(x\) is:

\[
p(\theta|x) = \frac{p(x|\theta) \cdot p(\theta)}{p(x)}
\]

If \(\theta\) is continuous, the marginal density (or predicted function) of the observed vector \(x\) is calculated as

\[
p(x) = \int_{\theta \in \Theta} p(x|\theta) \cdot \pi(\theta) \, d\theta
\]

In case of discrete \(\theta\),

\[
p(x) = \sum_{\theta \in \Theta} p(x|\theta) \cdot \pi(\theta)
\]

where \(p(x|\theta)\) is the likelihood function (LF). After the ensemble of sub-networks are trained, the issue of finding effective ways to combine the individual network outputs needs to be resolved.

In this paper, we propose to consider the combination of members of the ensemble as a statistical decision problem, in which each individual network is a decision-maker, the corresponding combination weight acts as the corresponding reliability measure. One of the advantages of Bayesian decision analysis is that it can model uncertain information via Bayesian reasoning process (Lampinen & Vehtari, 2001), which can help the analyst gain more insights into the system to be modeled. The proposed method consists of two steps: the first step is to train the networks of the ensemble; the second step is to learn the inter-connecting combination weights for the individual networks in a sequential fashion. At each stage, the available data samples plus the newly acquired data set are split into training data sets, test data sets, and validation data set. The training data sets are used to train the members of the ensemble, and the validation samples are used to identify the optimal combination weights in the second step. Then the generalization errors of each component network and the likelihood function (LF) value of each component network are calculated on the test samples. Finally the combination weights are adjusted according to Bayesian reasoning, in which the current weights are posterior and the previous weights are prior. Initial prior weights are set to be equal if we know nothing about the initial
weights. This sequential Bayesian learning scheme is summarized as follows.

Given the training set, validation set, the learning rate, the error limit, and the maximal number of iterations, we have the following steps for MNN learning:

Step 1. Train all the component networks until the error limit or the maximal number of iteration is achieved;

Step 2. Split the validation set into S parts, each one denoted as \( S = \{ E_{s1}, \ldots, E_{sS} \} \);

Step 3. For \( i = 1 \) to \( S \):

(Step 3a). Compute the LF value \( o_j(j = 1, \ldots, K) \) as

\[
o_j = \frac{1}{\sum_{k=1}^{K} 1/\text{sse}_k}
\]

(Step 3b). Update the combination weights by using Bayesian reasoning:

\[
w_f = \begin{cases} 
  w_f = w_f^{-1} & \text{if } \sum_{j=1}^{K} w_j^{-1} o_j = 0 \\
  w_f^{-1} o_j & \text{otherwise}
  \end{cases}
\]

From the (Step 3a), it can be seen that the validation data sets are constructed in a sequential way so that each validation set possesses certain property of inheritance. In the process of learning the combination weights, the (intra-network) connection weights of each component network do not change. Hence, in this sequential Bayesian learning algorithm, the global model performance is gradually improved.

### 3. Experimental results

In this section, eight benchmark regression examples (Zhou et al., 2002) are used to evaluate the proposed Bayesian learning algorithm for constructing MNNs, and compare with the learning scheme used in (Perrone & Cooper, 1994).

The eight benchmark regression examples are described in Table 1. For each problem, the proposed algorithm is run five times to reduce the potential randomness. The model performance is evaluated by averaging the five run results. First, we need to generate a data set and get the training set, validation set and test set. The training set is used to learn the connection weights of each component network, evaluation set to learn the combination weights, and test set to check the generalization performance of the whole system. The size of each data set and variables’ distribution interval are shown in Table 2, where \( x - y - z \) represents the size of these sets. Then the architecture of each individual network is constructed. Table 3 shows the architecture of each modular neural network, where \( x - y - z \) (or \( x - y_1 - y_2 - z \)) means the number of units in input, hidden and output layer are \( x, y \) (or \( y_1, y_2, \) and \( z \)). The activation function of the neurons in the hidden layer(s) is sigmoid function and the one in output layer is linear.

The next step is to train all the ensemble members of each MNN. Firstly, we split the training set into \( K \) (the number of the component networks) subsets. Then on each subset a component network is trained. The training parameters are indicated in Table 2 for each

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size of dataset</th>
<th>Partition of dataset</th>
<th>Number of component networks</th>
<th>Training error limit</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH2</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>4</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>MH3</td>
<td>4000</td>
<td>2000–1000–1000</td>
<td>5</td>
<td>1e-5</td>
<td>3000</td>
</tr>
<tr>
<td>FI</td>
<td>5000</td>
<td>3200–1000–800</td>
<td>8</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>F2</td>
<td>21,000</td>
<td>9000–10,000–2000</td>
<td>6</td>
<td>1</td>
<td>3000</td>
</tr>
<tr>
<td>GB</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>6</td>
<td>1e-4</td>
<td>5000</td>
</tr>
<tr>
<td>MI</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>8</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>PL</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>5</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>PO</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>4</td>
<td>1e-6</td>
<td>5000</td>
</tr>
</tbody>
</table>

### Table 3

Architecture of component networks.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Architecture of each component network</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH2</td>
<td>1-10-1, 1-10-1, 1-12-1, 1-12-1</td>
</tr>
<tr>
<td>MH3</td>
<td>2-8-8-1, 2-8-8-1, 2-8-10-1, 2-10-10-1</td>
</tr>
<tr>
<td>FI</td>
<td>5-18-1, 5-18-1, 5-20-1, 5-20-1, 5-22-1, 5-22-1, 5-24-1, 5-24-1</td>
</tr>
<tr>
<td>F2</td>
<td>4-25-1, 4-25-1, 4-30-1, 4-30-1, 4-35-1</td>
</tr>
<tr>
<td>GB</td>
<td>2-12-1, 2-12-1, 2-14-1, 2-6-8-1, 2-8-8-1, 2-8-8-1</td>
</tr>
<tr>
<td>MI</td>
<td>5-10-1, 5-10-1, 5-12-1, 5-12-1, 5-14-1, 5-16-1, 5-16-1</td>
</tr>
<tr>
<td>PL</td>
<td>2-8-1, 2-8-1, 2-10-1, 2-5-5-1, 2-5-5-1</td>
</tr>
<tr>
<td>PO</td>
<td>1-10-1, 1-10-1, 1-12-1, 1-12-1</td>
</tr>
</tbody>
</table>

### Table 1

Eight regression examples.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Variable(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-D Mexican Hat (MH2)</td>
<td>( y = \frac{\text{sin} \pi x_1 x_2}{1 + x_1^2 + x_2^2} )</td>
<td>( x \sim U[-2\pi, 2\pi] )</td>
</tr>
<tr>
<td>3-D Mexican Hat (MH3)</td>
<td>( y = \frac{\text{sin} \sqrt{x_1^2 + x_2^2}}{\sqrt{x_1^2 + x_2^2}} )</td>
<td>( x_1 \sim U[-4\pi, 4\pi] )</td>
</tr>
<tr>
<td>Friedman (F1)</td>
<td>( y = 10 \text{sin}(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] ) ( x_3 \sim U[0, 100] ) ( x_4 \sim U[0, 1] ) ( x_5 \sim U[0, 1] )</td>
</tr>
<tr>
<td>Friedman (F2)</td>
<td>( y = \sqrt{x_1^2 + x_2^2} ) ( z = \frac{1}{x_5} )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] ) ( x_3 \sim U[0, 1] ) ( x_4 \sim U[0, 1] ) ( x_5 \sim U[0, 1] )</td>
</tr>
<tr>
<td>Gabor (GB)</td>
<td>( y = \frac{1}{2} \exp[-(x_1^2 + x_2^2)] ) ( \cos[2\pi(x_1 + x_2)] )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] )</td>
</tr>
<tr>
<td>Multi-interaction (MI)</td>
<td>( y = 0.79 + 1.227x_1 x_2 + 1.569x_1 x_3 + 3.428x_2 x_3 + 2.06x_1 x_2 x_3 )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] ) ( x_3 \sim U[0, 1] )</td>
</tr>
<tr>
<td>Plane (PL)</td>
<td>( y = 0.6x_4 + 0.3x_5 )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] ) ( x_3 \sim U[0, 1] ) ( x_4 \sim U[0, 1] )</td>
</tr>
<tr>
<td>Polynomial (PO)</td>
<td>( y = 1 + 2x + 3x^2 + 4x^3 + 5x^4 )</td>
<td>( x_1 \sim U[0, 1] ) ( x_2 \sim U[0, 1] ) ( x_3 \sim U[0, 1] )</td>
</tr>
</tbody>
</table>

### Table 2

Generation of data sets.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Size of dataset</th>
<th>Partition of dataset</th>
<th>Number of component networks</th>
<th>Training error limit</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH2</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>4</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>MH3</td>
<td>4000</td>
<td>2000–1000–1000</td>
<td>5</td>
<td>1e-5</td>
<td>3000</td>
</tr>
<tr>
<td>FI</td>
<td>5000</td>
<td>3200–1000–800</td>
<td>8</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>F2</td>
<td>21,000</td>
<td>9000–10,000–2000</td>
<td>6</td>
<td>1</td>
<td>3000</td>
</tr>
<tr>
<td>GB</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>6</td>
<td>1e-4</td>
<td>5000</td>
</tr>
<tr>
<td>MI</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>8</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>PL</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>5</td>
<td>1e-6</td>
<td>5000</td>
</tr>
<tr>
<td>PO</td>
<td>3000</td>
<td>1200–1000–800</td>
<td>4</td>
<td>1e-6</td>
<td>5000</td>
</tr>
</tbody>
</table>
regression problem. Then the proposed sequential Bayesian learning algorithm is used to perform information aggregation. Table 4 shows the generalization performance of the proposed method in comparison with the scheme used in (Perrone & Cooper, 1994). It can be seen that the proposed algorithm outperforms the scheme used in (Perrone & Cooper, 1994) for aggregating the outputs of ensemble members of MNNs on the eight benchmark problems.

4. Conclusion

In this paper, a sequential Bayesian learning method is proposed for modular neural network to combine the outputs of independent component networks. At each stage the likelihood function (LF) value of each component network is computed first, then the combination weights are adjusted according to Bayesian reasoning, in which the current weights are posterior and the previous weights are prior. The results of experiments on the eight benchmark problems show that the proposed method outperforms the widely used scheme for MNN.

Acknowledgements

This research has been supported by the NSFC (National Natural Science Foundation of China) under the Grant 60174039, the research fund from the Institute of Systems Science and Engineering of Wuhan University of Technology, and the research fund from the Changjiang Scholar Program, Chinese Ministry of Education.

References


Table 4
Generalization performance comparison.

<table>
<thead>
<tr>
<th>MH2</th>
<th>MH3</th>
<th>F1</th>
<th>F2</th>
<th>GB</th>
<th>MI</th>
<th>PL</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.282e−7</td>
<td>1.717e−5</td>
<td>3.322e−5</td>
<td>3.91787</td>
<td>5.992e−6</td>
<td>0.000474</td>
<td>3.383e−7</td>
<td>5.987e−7</td>
</tr>
</tbody>
</table>

MH3 is (Perrone & Cooper, 1994) 6.734e−8

MH2 and MI are the generalization performance of the proposed method in comparison with the scheme used in (Perrone & Cooper, 1994). It can be seen that the proposed algorithm outperforms the scheme used in (Perrone & Cooper, 1994) for aggregating the outputs of ensemble members of MNNs on the eight benchmark problems.