Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A modular neural network architecture with concept

Yi Ding [a,b], Qi Feng [a,*], Tianjiang Wang [a], Xian Fu [b]

[a] Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[b] Department of Computer Science and Technology, Hubei Normal University, Huangshi 435002, China

ARTICLE INFO

Available online 26 February 2013

Keywords:
Neural network architecture
Modular neural network
Simulation
Generalization
Neural networks

ABSTRACT

This paper focuses on the powerful concept of modularity. It is descried how this concept is deployed in natural neural networks on an architectural as well as on a functional level. Furthermore, different approaches for modular neural networks are discussed. By means of these methods, a two-layer modular neural system is introduced. The basic building blocks of the architecture are multilayer perceptions (MP) with the backpropagation (BP) algorithm. This modular network is designed to combine two different approaches of generalization known from connectionist and logical neural networks; this enhances the generalization abilities of the network. Experiments described in this paper show that the architecture is especially useful in solving problems with a large number of input attributes.

1. Introduction

Artificial Neural Networks (ANNs) are of major research interest at present, involving researchers of many different disciplines. Subjects contributing to this research include biology, computing, electronics, mathematics, medicine, physics and psychology [1]. The approaches to this topic are very diverse, as are the aims. The basic idea is to use the knowledge of the nervous system and the human brain to design intelligent artificial systems [2].

On one hand, biologists and psychologists are trying to model and understand the brain and parts of the nervous system and searching for explanations for human behavior and reasons for the limitations of the brain [3]. On the other hand, computer scientists and electronic engineers are searching for efficient ways to solve problems for which conventional computers are currently used. Biological and psychological models and ideas are often the resource of inspiration for these scientists [4].

2. Modularity in artificial neural networks

Modularity is a very important concept in nature. Modularity can be defined as subdivision of a complex object into simpler objects [5]. The subdivision is determined either by the structure or function of the object and its subparts [6].

Replication and decomposition are the two main concepts for modularity. These concepts are found in concrete objects as well as in thinking. It is often difficult to discriminate sharply between them; replication and decomposition often occur in combination [7].

Replication is a way of reusing knowledge. Once one module is developed and has proved to be useful it is replicated in a larger number [8]. Decomposition is often found when dealing with a complex task. It is a sign of intelligent behavior to solve a complex problem by decomposing it into simpler tasks which are easier to manage and then reassemble the solution from the results of the subtasks. Constructing large software, building a car, or solving an equation is usually done by decomposing the problem [9].

A modular neural system using a self-organizing map and a multilayer perception is presented in [10]. It is applied in a cosmic ray space experiment. Many other researchers have also investigated the concept of modularity and the impact on neural networks.

In the following section a modular neural network is proposed to enhance the generalization ability of neural networks for high dimensional inputs. The network consists of several MPs. Each of the modules is trained by the BP algorithm. The number of weight-connections in the proposed architecture is significantly smaller than in a comparable monolithic network [11].

3. A new modular neural network

The aim of the project was to develop a parallel and fault tolerant modular network architecture. The focus on parallelism and fault tolerance was motivated by the structure of the human nervous system.

3.1. The network architecture

The proposed network system consists of a layer of input modules and an additional decision module. All sub-networks are

* Corresponding author.
E-mail addresses: teacher.dingyi@yahoo.com.cn (Y. Ding), a_carrie@sina.com (Q. Feng), wt_jiangmail@yahoo.com.cn (T. Wang), teacher.fu@yahoo.com.cn (X. Fu).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2012.06.051

MPs. Only the number of inputs and the number of outputs of the module is determined by the system. The internal structure, such as the number of hidden layers and the number of neurons in each hidden layer can be chosen independent of the overall architecture.

Each input variable is connected to only one of the input modules. These connections are chosen at random. The outputs of all input modules are connected to the decision network. In the discussion that follows the dimension of the input vector is denoted by l and the number of classes by k.

To determine a modular network it is necessary to specify either the number of inputs per input module or the number of input modules. These parameters are dependent on each other. It is assumed here that the number of inputs per module in the first layer is chosen as n. The number of input modules in the input layer can therefore be calculated as $m = \lceil l/n \rceil$.

It is further assumed that $l = m \cdot n$. If this is not the case the spare inputs can be connected to constant inputs; in the implementation of the model all 'free' inputs were connected to the constant value '0'. Alternatively, it would be possible to alter the size of one module or of a group of modules.

Each module in the input layer can have either k or $\lceil \log_2 k \rceil$ outputs. The network with k intermediate outputs is referred to as large intermediate representation. It is useful only if the number of classes is very small. For problems with a larger number of classes the small intermediate representation ($\lceil \log_2 k \rceil$) is more appropriate.

From an information theory point of view the small intermediate representation should be sufficient because only this number of output neurons is required to represent all the classes in a binary code.

The decision network has $(m \cdot n)$ or $m \cdot \lceil \log_2 k \rceil$) inputs, dependent on the intermediate representation used. The number of outputs is k, one output neuron for each class. The structure of the modular network is depicted in Fig. 1 using a small intermediate representation. The function $\pi : X \mapsto X$ gives a permutation on $X = \{1 \ldots l\}$. This permutation is randomly chosen and constant for a network.

Definition 1 (*A module*). A module is a multilayer feedforward neural network defined by a 3-tuple:

$$M = (a, b, H),$$

where a is the number of inputs of the module, b is the number of output nodes and H is a list containing the numbers of neurons in each of the hidden layers.

Definition 2 (*A modular neural network*). A modular neural network is a set of interconnected modules defined by a 7-tuple:

$$N = (l, k, m, r, \pi, L, D),$$

where l is the number of inputs, k is the number of classes, m is the number of modules in the input layer, r is the type of the intermediate representation $r \in \{small, large\}$, π is the permutation function, L is the input layer module and D is the decision module.

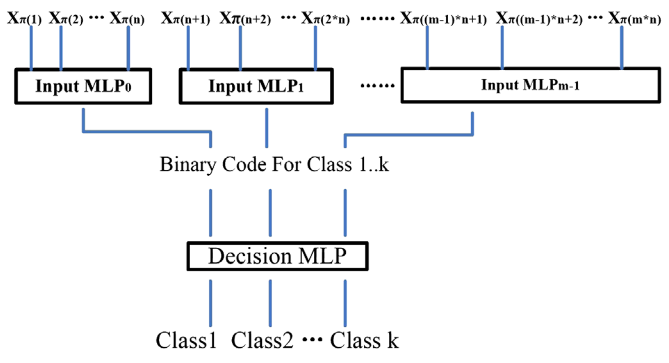
Xπ(1) Xπ(2) ⋯ Xπ(n)
Xπ(n+1) Xπ(n+2) ⋯ Xπ(2*n)
⋯⋯ Xπ((m-1)*n+1) Xπ((m-1)*n+2) ⋯ Xπ(m*n)
Input MLP0
Input MLP1
⋯⋯
Input MLPm-1
Binary Code For Class 1..k
Decision MLP
Class1 Class2 ⋯ Class k

Fig. 1. The modular neural network architecture.

3.2. Training the system

Training occurs in two stages, using the Back propagation algorithm.

In the first phase all sub-networks in the input layer are trained. The individual training set for each sub-network is selected from the original training set and consists of the components of the original vector which are connected to this particular network (as an input vector) together with the desired output class represented in binary or l-out-of-k coding.

In the second stage the decision network is trained. To calculate the training set each original input pattern is applied to the input layer; the resulting vector together with the desired output class (represented in a l-out-of-k coding) form the training pair for the decision module.

To simplify the description of the training a small intermediate representation is used, further it is assumed that the permutation function is the identity $\pi(x) = x$. The original training set TS is $(x_1^j, x_2^j, \ldots, x_l^j; d^j)$, where $x_i^j \in R$ is the ith component of the jth input vector, d^j is the class number and $j = 1, \ldots, t$, where t is the number of training instances. The module MP_i is connected to

$$x_{i \cdot n+1}, x_{i \cdot n+2}, \ldots, x_{(i+1) \cdot n}.$$

The training set TS_i for the module MP_i,

$$(x_{i \cdot n+1}^j, x_{i \cdot n+2}^j, \ldots, x_{i \cdot n+n}^j; d_{BIN}^j),$$

for all $j = 1, \ldots, t$, where d_{BIN}^j is the output class d^j represented in a binary code. The mapping performed by the input layer is denoted by

$$\Phi : R^{n \cdot m} \mapsto R^{m \cdot \lceil \log_2 k \rceil}.$$

The training set for the decision network, $(\Phi(x_1^j, x_2^j, \ldots, x_l^j); d_{BIT}^j)$ and $j = 1, \ldots, t$, where d_{BIT}^j is the output class d_j represented in a l-out-of-k code. The mapping of the decision network is denoted by

$$\Psi : R^{m \cdot \lceil \log_2 k \rceil} \mapsto R^k.$$

The training of each module in the input layer is independent of all other modules so this can be done in parallel. The training is stopped either when each module has reached a sufficient small error or a defined maximum number of steps has been performed. This keeps the modules independent.

3.3. Calculation of the output

The calculation of the output also occurs in two stages. First the input sub-vectors for each module are selected from the applied input vector according to the permutation function and the intermediate output is calculated by all modules. In the second stage all the outputs from the input layer are used as input to the decision network, then the final result is computed.

The mapping of the whole network is denoted by

$$\Phi \circ \Psi : R^l \mapsto R^k.$$

The response r for a given test input $(a_1, a_2, \ldots, a_l)$ is determined by the following function:

$$r = \Psi(\Phi(a_1, a_2, \ldots, a_l)).$$

The k-dimensional output of the decision module is used to determine the class number for the given input. In the experiments the output neuron with the highest response was chosen as the calculated class. The differences between this neuron and the runner-up may be taken as a measure of accuracy.