Knowledge-Based Systems 74 (2015) 159–166

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

# A local learning algorithm for random weights networks

Jianwei Zhao [a], Zhihui Wang [a], Feilong Cao [a,*], Dianhui Wang [b,c]

[a] Department of Information and Mathematics Sciences, China Jiliang University, Hangzhou 310018, Zhejiang Province, PR China
[b] Department of Computer Science and Computer Engineering, La Trobe University, Melbourne, Victoria 3086, Australia
[c] The State Key Laboratory of Synthetical Automation for Process Industries (Northeast University), Shenyang 110819, Liaoning Province, PR China

ARTICLE INFO

*Article history:*
Received 23 August 2014
Received in revised form 5 November 2014
Accepted 12 November 2014
Available online 20 November 2014

*Keywords:*
Random weights networks
Local learning algorithm
Regularization model
Moving least squares method
Feedforward neural networks

ABSTRACT

Robust modelling is significant to deal with complex systems with uncertainties. This paper aims to develop a novel learning algorithm for training regularized local random weights networks (RWNs). The learner model, terms as RL-RWN, is built on regularized moving least squares method and generalizes the solution obtained from the standard least square technique. Simulations are carried out using two benchmark datasets, including Auto-MPG data and surface reconstruction data. Results demonstrate that our proposed RL-RWN outperforms the original RWN and radial basis function networks.

## 1. Introduction

Feedforward Neural Networks (FNNs) are often used in data regression and classification with applications in various domains. In conventional learning techniques, all parameters (e.g., weights and bias) of a neural network are required freely adjustable, that is, they can be tuned properly and the appropriate values can be found from some sample data. For example, the well-known error back-propagation (BP) algorithm [1], adjusts the parameters of FNNs with gradient descent method.

However, the speed of the BP algorithm is terribly slow and often suffers from local minima that is related with the choice of initial weights. If the hidden weights and biases are chosen randomly, i.e., they are considered as random variables with the uniform distribution on $(0,1)$, then the output weights of FNNs can be determined by solving a least square problem with respect to the output weights. The original idea of building neural networks with random weights (RWNs) can be found in [2] and later on such a random learner model was further investigated with a universal approximation theorem [3,4].

It has been pointed out in [5,6] that FNNs have limited ability to characterize local features, such as discontinuities in curvature, jumps in value or other edges. Those local features, which are located in time and/or frequency, typically embody important process-critical information such as aberrant process modes or faults [5]. Bottou has suggested an improved localized models in [7] to deal with both data reduction (or compression) and subsequent classification tasks, that rely on an accurate representation of local features.

Zhang [8], Bakshi and Stephanopoulos [9], Hou and Han [10], Zainuddin and Pauline [11] have developed Wavelet Neural Networks (WNNs), and tried to overcome this weakness of FNNs by means of using the local representation property of wavelets. However, like FNNs and RWNs, the objective function for training WNNs is the Mean Square Error (MSE), i.e., the sum of square error between the observed values and the predicted values. In this way, each training datum has equal weight in the square error, so they cannot reflect the different influence of training data on testing data. Even for the weighted FNNs [12], each training datum has different weight in the square error, the proportions are fixed in the process of learning. That is, they do not change with the variation of testing data. While in many real world applications such as surface reconstruction and signal processing, the training data that are closer to the testing data should have more influence than others. That is, the weights (proportions) should be updated according to the closeness of the testing data to teacher signals.

In this paper, we embody the local features of the models into learning and utilization processes of the RWN, called regularized local random weights networks (RL-RWNs). The proposed learning algorithm can reflect the different influence of training data on each testing data that can be understood as follows: For each testing sample, firstly, select a few data from the training dataset located in the vicinity of the testing datum; Secondly, train an

* Corresponding author.

http://dx.doi.org/10.1016/j.knosys.2014.11.014
0950-7051/© 2014 Elsevier B.V. All rights reserved.

FNN with only these selected training data, and finally apply the learned network to predict the outputs of the testing data.

The proposed RL-RWN will improve the performances of RWN with the modified Moving Least Squares (MLS) method based on regularization model. The MLS method, proposed by Lancaster and Salkauskas in [13], represents the local approximation by adding some weighted functions that character the influence of training data on testing data in the square error. But in the MLS method, the basis are assumed to be independent over the set of inputs of training data, which guarantees the existence and uniqueness of the solution. However, in the proposed RL-RWN with random hidden-layer weights and biases, may result in an ill-posed linear system. In this case, we employ the well-known regularization model in our algorithm development. It has been observed that with such a regularization loss function and local modelling concept, the generalization of the RWN has been greatly improved. Experimental comparisons of RL-RWN with Radial Basis Function (RBF) networks and RWN on Auto-MPG and surface reconstruction datasets are quite supportive.

The remainder of the paper is organized as follows. Section 2 gives a brief review of RWNs. Section 3 describes a new local learning algorithm for training RWNs based on an improved MLS method. Section 4 reports experimental results with comparisons. Section 5 concludes this paper.

## 2. Brief review of RWN

Some basic concepts and related works about RWN are briefly reviewed in this section to provide a background for the proposed RL-RWN in Section 3.

The output of an FNN with $L$ hidden-layer nodes (additive or RBF nodes) can be represented by

$$f(\mathbf{x}) = \sum_{i=1}^{L}\beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \tag{1}$$

where $\mathbf{a}_i \in \mathbb{R}^n$ and $b_i \in \mathbb{R}$ are the input weights and bias of hidden-layer nodes, respectively, $\beta_i \in \mathbb{R}^m$ is the output weight, and $G(\mathbf{a}_i, b_i, \mathbf{x})$ denotes the output of the $i$-th hidden-layer node with respect to the input $\mathbf{x}$. Additive and RBF hidden-layer nodes are often used in applications.

For a given data set $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \mathbb{R}^n \times \mathbb{R}^m$, where $\mathbf{x}_i$ is an $n \times 1$ input vector and $\mathbf{t}_i$ is the corresponding $m \times 1$ observed vector, an FNN with $L$ hidden-layer nodes approximating these $N$ training data with zero error means that there exist $\mathbf{a}_i$, $b_i$ and $\beta_i$ $(i = 1, 2, \ldots, L)$ such that

$$\sum_{i=1}^{L}\beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j + \varepsilon_j, \quad j = 1, 2, \ldots, N, \tag{2}$$

where $\varepsilon_j$ is the measure error. The system (2) can be rewritten in the following compact form:

$$\mathbf{H}\beta = \mathbf{T} + \varepsilon, \tag{3}$$

where

$$\mathbf{H} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N\times L}, \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^\top \\ \vdots \\ \beta_L^\top \end{bmatrix}_{L\times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^\top \\ \vdots \\ \mathbf{t}_N^\top \end{bmatrix}_{N\times m} \text{and} \quad \varepsilon = \begin{bmatrix} \varepsilon_1^\top \\ \vdots \\ \varepsilon_N^\top \end{bmatrix}_{N\times m}. \tag{5}$$

Here, $\beta^\top$ denotes the transpose of the vector $\beta$, $\mathbf{H}$ is the hidden-layer output matrix with respect to the input vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$.

According to RWN, the input weights $\mathbf{a}_i$ and $b_i$ need not to be adjusted during the process of learning and can be assigned with random values. Then, Eq. (3) becomes a linear equation system and the output weight $\beta$ can be obtained directly, that is,

$$\hat{\beta} = \mathbf{H}^\dagger\mathbf{T}, \tag{6}$$

where $\mathbf{H}^\dagger$ is the Moore–Penrose pseudo inverse of the hidden-layer output matrix $\mathbf{H}$ [14]. There are several methods to calculate the Moore–Penrose pseudo inverse of a matrix, including orthogonal projection method, orthogonalization method, iterative method and singular value decomposition. If $\mathbf{H}^\top\mathbf{H}$ is nonsingular, then $\mathbf{H}^\dagger = (\mathbf{H}^\top\mathbf{H})^{-1}\mathbf{H}^\top$.

**Algorithm RWN.** Given a training data set $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subset \mathbb{R}^n \times \mathbb{R}^m$, choose the hidden-node output function $G(\mathbf{a}, b, \mathbf{x})$ and the hidden-node number $L$.

**Step 1.** Randomly assign the input parameters $(\mathbf{a}_i, b_i)$, $i = 1, 2, \ldots, L$.
**Step 2.** Compute the hidden-layer output matrix $\mathbf{H}$ as in (4).
**Step 3.** Calculate the output weight $\hat{\beta} = \mathbf{H}^\dagger\mathbf{T}$.

## 3. Proposed algorithm RL-RWN

In many applications, training data often have their own influential strength. Although weighted learning algorithms can embody varying proportions in the square error, the proportions are fixed in the process of learning. That is, they do not change with the variation of testing data. Therefore, it cannot effectively reflect the local features of models. In order to address the influences of training data on the testing performance, we propose a novel learning algorithm by means of the MLS method and a regularization model.

In order to embody the influences of the training data on the testing samples, a new objective function and some notation are introduced [15,16].

### *3.1. Notation*

Let $(\mathbb{R}^m)^N$ be the direct sum of $N$ Hilbert space $\mathbb{R}^m$, then it is a real Hilbert space [15]. The inner product $\langle,\rangle_N$ on $(\mathbb{R}^m)^N$ is defined as

$$\langle \mathbf{x}, \mathbf{y}\rangle_N = \sum_{i=1}^{N}\langle \mathbf{x}_i, \mathbf{y}_i\rangle = \sum_{i=1}^{N}\sum_{j=1}^{m} x_{ij}y_{ij} \tag{7}$$

and the induced norm on $(\mathbb{R}^m)^N$ by $\langle,\rangle_N$ is given by

$$\|\mathbf{x}\|_N = \left(\sum_{i=1}^{N}\sum_{j=1}^{m} x_{ij}^2\right)^{\frac{1}{2}}, \tag{8}$$

where $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N] \in (\mathbb{R}^m)^N$ and $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^\top$, $\mathbf{y}_i = [y_{i1}, y_{i2}, \ldots, y_{im}]^\top \in \mathbb{R}^m$, $i = 1, 2, \ldots, N$.

For a given training set $\{(\mathbf{x}_i, \mathbf{t}_i)\}_{i=1}^{N} \subseteq D \times \mathbb{R}^m$ and a set of weighted functions $\{w_i(\cdot)\}_{i=1}^{N}$ from $\mathbb{R}^n$ to $\mathbb{R}^+$, where $D$ is a bounded domain of $\mathbb{R}^n$, we can construct a new norm $\|\cdot\|_\mathbf{z}$ with respect to the point $\mathbf{z} \in D$ on the direct sum $(\mathbb{R}^m)^N$ of $N$ Hilbert space $(\mathbb{R}^m, \langle,\rangle)$ to give a description of objective function.

Let

$$W(\mathbf{z}) = \mathrm{diag}(w_1(\mathbf{z}), w_2(\mathbf{z}), \ldots, w_N(\mathbf{z})) \tag{9}$$

be the weighted matrix with respect to the variable $\mathbf{z}$, then for any two vectors $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ and $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]$ in $(\mathbb{R}^m)^N$, we can define an inner product $\langle,\rangle_\mathbf{z}$ on $(\mathbb{R}^m)^N$ as follows:

$$\langle \mathbf{u}, \mathbf{v}\rangle_\mathbf{z} = \sum_{i=1}^{N} w_i(\mathbf{z})\langle \mathbf{u}_i, \mathbf{v}_i\rangle, \tag{10}$$