Neurocomputing 171 (2016) 63–72

Contents lists available at ScienceDirect

Neurocomputing

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

A new deep neural network based on a stack of single-hidden-layer feedforward neural networks with randomly fixed hidden neurons

Junying Hu, Jiangshe Zhang*, Chunxia Zhang, Juan Wang

School of Mathematics and Statistics, Xi'an Jiaotong University, China

ARTICLE INFO

Article history:
Received 29 January 2015
Received in revised form
9 June 2015
Accepted 9 June 2015
Communicated by S. Mitra
Available online 2 July 2015

Keywords:
Single-hidden layer feedforward neural network
Manifold regularization
Unsupervised learning
Embedding
Stackable structure

ABSTRACT

Single-hidden layer feedforward neural networks with randomly fixed hidden neurons (RHN-SLFNs) have been shown, both theoretically and experimentally, to be fast and accurate. Besides, it is well known that deep architectures can find higher-level representations, thus can potentially capture relevant higher-level abstractions. But most of current deep learning methods require a long time to solve a non-convex optimization problem. In this paper, we propose a stacked deep neural network, St-URHN-SLFNs, via unsupervised RHN-SLFNs according to stacked generalization philosophy to deal with unsupervised problems. Empirical study on a wide range of data sets demonstrates that the proposed algorithm outperforms the state-of-the-art unsupervised algorithms in terms of accuracy. On the computational effectiveness, the proposed algorithm runs much faster than other deep learning methods, i.e. deep autoencoder (DA) and stacked autoencoder (SAE), and little slower than other methods.

1. Introduction

Single-hidden layer feedforward networks (SLFNs) have been intensively studied during the past several decades. However, most of the existing learning algorithms for training SLFNs basically introduce high computational cost. Recently, Huang et al. [1] proposed the extreme learning machines (ELMs) for training SLFNs, which construct the hidden layer using a fixed number of randomly generated mapping neurons and analytically determines the output weights of SLFNs by minimizing the sum of the squared losses of the prediction errors. It has been shown that ELMs are much more efficient and usually lead to better generalization performance than traditional methods for SLFNs.

Actually, the idea of ELMs that hidden neurons are randomly assigned and the output weights only need to be trained for a SLFN has been introduced by other researchers in the early stages. For example, Schmidt et al. [2] introduced feedforward neural networks with random weights where the weights of hidden layer (s) were chosen randomly, whereas the output layer trained by a single layer learning rule or a pseudo-inverse technique. The minor difference is that the bias of the output neuron in ELM is set to zero, while the bias of the output neuron in [2] is not set to zero. However, there is nothing preventing the bias of the output neuron in [2] to assume a zero value. Pao et al. [3] proposed the random vector function-link network (RVFL) where the hidden neurons in an function-link neural network (FLN) were randomly selected and only the weights of the output layer needed to be trained, with the only difference from the ELMs being that RVFL allows for direct connections from the input nodes to the output neurons, whereas the ELM dose not. Another related work is radial basis function (RBF) neuron networks with randomly selected RBF centers and suitably selected RBF width which is proposed by Broomhead and Lowe in their classic paper [4,5]. We can see that RBF network in [4,5] is a minor variation of traditional RBF network, since Park and Sandberg [6] proved that RBF networks with either the same width for all RBF neurons in the network or different widths for different RBF neurons in the network are universal approximators. Actually, Huang et al. [7–9] have stated the differences between ELMs and other related works. Meanwhile, Hornik [10] showed that a single hidden layer feedforward network with arbitrary bounded and nonconstant activation function are universal approximation. Based on the above discussed, we refer the above all sorts of SLFNs as SLFNs with randomly fixed hidden neurons (RHN-SLFNs).

Motivated by unsupervised extreme learning machines [11] which extending ELM for unsupervised tasks based on the manifold regularization and the great success of deep learning [12,13], we propose deep neural network using a stack of unsupervised RHN-SLFNs (St-URHN-SLFNs). The proposed model not only incorporates the simplicity of RHN-SLFNs but also possesses the power derived from deep architectures. The proposed St-URHN-SLFNs is a

* Corresponding author.
E-mail addresses: hujunyingmm@163.com (J. Hu), jszhang@mail.xjtu.edu.cn (J. Zhang), cxzhang@mail.xjtu.edu.cn (C. Zhang), Wangjuan03022204@163.com (J. Wang).

http://dx.doi.org/10.1016/j.neucom.2015.06.017
0925-2312/© 2015 Elsevier B.V. All rights reserved.

simple stack of unsupervised RHN-SLFNs's, namely, utilizing unsupervised RHN-SLFNs as the base building block. Like the greedy, layer-by-layer unsupervised learning algorithm introduced by Hinton et al. [14], the proposed algorithm consists of learning a stack of unsupervised RHN-SLFNs one layer at a time. After the stack of the base building block, the whole stack can be viewed as a single deep neural network model, called a St-URHN-SLFN. We test our algorithms on a variety of data sets and the results show that the proposed algorithms outperform the unsupervised RHN-SLFNs algorithm and other state-of-the-art algorithms in terms of accuracy. On the computational effectiveness, the proposed algorithm runs much faster than deep autoencoder (DA) and stacked autoencoder (SAE), and little slower than other methods.

The rest of the paper is organized as follows. In Section 2 we give a brief review of RHN-SLFNs and manifold regularization. Section 3 gives a brief review of unsupervised RHN-SLFNs. Section 4 introduces the basic model structure of St-URHN-SLFNs and gives specific algorithm description. Section 5 presents experimental results. Section 6 concludes this paper.

2. Preliminaries

In this section, in order to have a better understanding of unsupervised RHN-SLFNs, RHN-SLFNs and manifold regularization framework are introduced.

2.1. Introduction of RHN-SLFNs

RHN-SLFNs is the simplest kind of neural network. It consists of three layers: an input layer, a hidden layer and an output layer. The output of each hidden unit is determined by forming a weighted sum of unit values in the input layer and then passing this result through an arbitrary bounded and nonconstant function which is usually a sigmoidal function or Gaussian function (we use sigmoidal function, e.g. $g(x)=1/(1+e^{-x})$, in this paper). The output of each output unit is a weighted sum of unit values in the hidden layer. For presentation purpose, we give the following explanation of symbols.

- n_i: the number of input units;
- n_o: the number of output units;
- n_h: the number of hidden units;
- $\mathbf{W}_{n_i \times n_h}$: the input weights between the input layer and the hidden layer which is $n_i \times n_h$ matrix;
- $\mathbf{b}$: the biases of hidden units;
- $\boldsymbol{\beta}$: the output weights between the hidden layer and the output layer which is $n_h \times n_o$ matrix.

The basic idea for training RHN-SLFNs is very simple, which only needs to update the output weights $\boldsymbol{\beta}$ by adopting the squared loss of the prediction error, while the parameters, i.e., the input weights $\mathbf{W}$ and biases of the hidden layer $\mathbf{b}$, are randomly generated. For N training samples $\{\mathbf{X},\mathbf{Y}\}=\{\mathbf{x}_i,\mathbf{y}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^{n_i}$, $\mathbf{y}_i$ is a n_o-dimensional binary vector with only one entry (correspond to the class that $\mathbf{x}_i$ belongs to) equal to one for multiclassification tasks, or $\mathbf{y}_i \in \mathbb{R}^{n_o}$ for regression tasks, the mathematical model for training RHN-SLFNs is as follows:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{n_h \times n_o}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{N}\|\boldsymbol{e}_i\|^2, \qquad \text{s.t.} \quad \boldsymbol{h}(\mathbf{x}_i)\boldsymbol{\beta} = \mathbf{y}_i^T - \boldsymbol{e}_i^T, \quad i=1,\dots,N, \tag{1}$$

where $\boldsymbol{h}(\mathbf{x}_i)=g(\mathbf{x}_i\mathbf{W}+\mathbf{b})$ is the output vector of the hidden layer with respect to $\mathbf{x}_i$, $\boldsymbol{e}_i \in R^{n_o}$ is the error vector with respect to the ith training sample, C is a penalty coefficient on the training errors. The model structure of RHN-SLFNs is shown in Fig. 1.

We substitute the constraints into the objective function to obtain the following equivalent unconstrained optimization problem:

$$\min_{\boldsymbol{\beta} \in \mathbb{R}^{n_h \times n_o}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\mathbf{Y}-\boldsymbol{H}\boldsymbol{\beta}\|^2 \tag{2}$$

where $\boldsymbol{H}=[\boldsymbol{h}(\mathbf{x}_1)^T,\dots,\boldsymbol{h}(\mathbf{x}_N)^T]^T \in \mathbb{R}^{N \times n_h}$.

In order to get the optimal solution of the above problem, we set the gradient of the above objective function with respect to $\boldsymbol{\beta}$ to zero and obtain the following equation:

$$\boldsymbol{\beta} - C\boldsymbol{H}^T(\mathbf{Y}-\boldsymbol{H}\boldsymbol{\beta}) = 0 \tag{3}$$

The solution of the above Eq. (3) is discussed in two cases. When the number of training patterns is larger than the number of the hidden neurons, $\boldsymbol{H}$ has more rows than columns. In this case, Eq. (3) is overdetermined, and we obtain the following closed form solution for (2):

$$\boldsymbol{\beta}^* = \left(\boldsymbol{H}^T\boldsymbol{H} + \frac{\boldsymbol{I}_{n_h}}{C}\right)^{-1}\boldsymbol{H}^T\mathbf{Y} \tag{4}$$

where $\boldsymbol{I}_{n_h}$ is an identity matrix of dimension n_h.

When the number of training patterns is smaller than the number of hidden neurons, $\boldsymbol{H}$ will have more columns than rows. In this case, the above Eq. (3) is underdetermined and $\boldsymbol{\beta}$ may have infinite number of solutions. To solve the problem, we introduce additional constraints to $\boldsymbol{\beta}:\boldsymbol{\beta}=\boldsymbol{H}^T\boldsymbol{\alpha}$ ($\boldsymbol{\alpha} \in \mathbb{R}^{N \times n_o}$). Notice that when $\boldsymbol{H}$ has more columns than rows and is row full rank, then $\boldsymbol{H}\boldsymbol{H}^T$ is invertible. Both sides of (3) are multiplied by $(\boldsymbol{H}\boldsymbol{H}^T)^{-1}\boldsymbol{H}$, we get

$$\boldsymbol{\alpha} - C(\mathbf{Y}-\boldsymbol{H}\boldsymbol{H}^T\boldsymbol{\alpha}) = 0 \tag{5}$$

Then the solution for (2) is given by

$$\boldsymbol{\beta}^* = H^T\boldsymbol{\alpha}^* = \boldsymbol{H}^T\left(\boldsymbol{H}\boldsymbol{H}^T + \frac{\boldsymbol{I}_N}{C}\right)^{-1}\mathbf{Y} \tag{6}$$

where $\boldsymbol{I}_N$ is an identity matrix of dimension N.

Based on the above discussion, we summarized the algorithm for training RHN-SLFNs as Algorithm 1.

Algorithm 1. Training algorithm of RHN-SLFNs.

```
Input:
  Labeled patterns, $\{\mathbf{X},\mathbf{Y}\}=\{\mathbf{x}_i,\mathbf{y}_i\}_{i=1}^N$;
output:
  The mapping function: $\boldsymbol{f}:\mathbb{R}^{n_i}\to\mathbb{R}^{n_o}$;
  Step 1: Initiate an RHN-SLFNs of $n_h$ hidden neurons with
    random input weights and biases.
  Step 2:
  • If $n_h \le N$
```

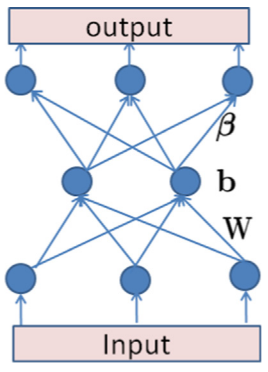

Fig. 1. Initializing the inputweights $\mathbf{W}$ and the biases of hidden layer $\mathbf{b}$ at random, the output of hidden layer $\boldsymbol{h}(\mathbf{x}_i)$ can be computed by $\boldsymbol{h}(\mathbf{x}_i)=g(\mathbf{x}_i\mathbf{W}+\mathbf{b})$, $i=1,2,\dots,N$. Then we can gain the outputweights β by solving (1).