Expert Systems with Applications 36 (2009) 12049–12053

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Improving the generalization performance of RBF neural networks using a linear regression technique

C.L. Lin [a], J.F. Wang [a], C.Y. Chen [b,d], C.W. Chen [c], C.W. Yen [a,*]

[a] Department of Mechanical and Electro-Mechanical Engineering, National Sun Yat-Sen University, Kaohsiung 80441, Taiwan
[b] Department of Management Information System, Yung-Ta Institute of Technology and Commerce, 316 Jong Shan, Rd., Lin Luoh, Pingtung 90941, Taiwan
[c] Department of Logistics Management, Shu-Te University, Kaohsiung 82445, Taiwan
[d] Department of Computer Science, National Pingtung University of Education, No. 4-18, Ming Shen Rd., Pingtung 90003, Taiwan

ARTICLE INFO

Keywords:
Radial basis function
Neural network
Function approximation
Generalization performance
Orthogonal least squares

ABSTRACT

In this paper we present a method for improving the generalization performance of a radial basis function (RBF) neural network. The method uses a statistical linear regression technique which is based on the orthogonal least squares (OLS) algorithm. We first discuss a modified way to determine the center and width of the hidden layer neurons. Then, substituting a QR algorithm for the traditional Gram–Schmidt algorithm, we find the connected weight of the hidden layer neurons. Cross-validation is utilized to determine the stop training criterion. The generalization performance of the network is further improved using a bootstrap technique. Finally, the solution method is used to solve a simulation and a real problem. The results demonstrate the improved generalization performance of our algorithm over the existing methods.

1. Introduction

Radial basis function neuron networks (RBF) have recently attracted extensive research interest (Huang, Su, & Kuo, 2008; Kurt, Mevlut, & Kurum, 2008; Lee, 2007; Schwenker, Kestler, & Plam, 2001). RBF can be regarded as an amalgamation of a data modeling technique for high-dimensional space, and a universal approximation scheme, also popularly known as artificial neural networks (ANNs). RBF neural networks are a powerful technique for generating multivariate nonlinear mapping (Bishop, 1991).

An RBF network approximates an unknown mapping function such as $f : \boldsymbol{R}^n \to \boldsymbol{R}^m$. First, the input data undergo a nonlinear transformation via the basis function in the network's hidden layer; the combined basis function responses are linearly combined to give the network output. The overall input–output transfer function of the RBF network can be defined and mapped as follows:

$$f_i(\mathbf{x}) = \hat{y}_i = \sum_{j=1}^{J} \phi_j(\|\mathbf{x} - \mathbf{c}_j\|, \rho) \cdot \theta_{ji}, \quad i = 1, \ldots, m, \tag{1}$$

where θ_{ji} indicates the adjustable network weights connecting hidden network nodes with network output; $\phi_j(\|\mathbf{x} - \mathbf{c}_j\|, \rho)$ is the activation function of the hidden layer neurons or the hidden kernel nodes in an RBF network. Typically this is a Gaussian function, as in Eq. (2). The $\mathbf{x}$ is the input variable; $\mathbf{c}$ is the center of the activation function; ρ is the width of the activation function; and $\|\cdot\|$ denotes the Euclidean norm

$$\phi_j(\|\mathbf{x} - \mathbf{c}_j\|, \rho) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_j\|^2}{\rho^2}\right). \tag{2}$$

As is well known, the performance of an RBF network is critically dependent on the number and the centers of the hidden layer neurons chosen. The most natural choice is to set each piece of data in the training set to correspond to a center (Bishop, 1991; Chen & Billings, 1992). This means that all the training data are center candidates. In this case, the number of degrees of freedom in the network is equal to the number of training data, and the network function fits each data point exactly. If the data behave regularly, but are contaminated by noise, then the phenomenon of overfitting is said to occur, which leads to poor generalization performance of the network. An efficient approach for improving generalization performance is to construct a small network using the parsimonious principle (Chen & Billings, 1992; Kavli, 1993). However, the parsimonious principle strategy is not entirely immune to overfitting (Chen & Billings, 1992). Regularization is a technique that can be used to overcome overfitting (Bishop, 1991). Some researchers have combined regularization techniques with the parsimonious principle. For example, Barron and Xiao (1991) proposed a first-order regularized stepwise selection of subset regression models. (Orr, 1993) derived a regularized forward selection algorithm. (Chen & Billings, 1992) combined zero-order regularization with the orthogonal least squares (OLS) method to derive a regularized OLS algorithm (ROLS) for RBF networks.

* Corresponding author. Tel.: +886 938433766.
E-mail addresses: cwchen@mail.stu.edu.tw, vincen@mail.nsysu.edu.tw (C.W. Yen).

0957-4174/$ - see front matter © 2009 Published by Elsevier Ltd.
doi:10.1016/j.eswa.2009.03.012

In this study we present a method for improving the generalization ability of RBF neural network by using a statistics linear regression technique based on the OLS algorithm. We first discuss a modified way to determine the center and width of the hidden layer neurons. Then, substituting the QR algorithm for the traditional Gram–Schmidt algorithm, we solve for the connected weight of the hidden layer neurons. Cross-validation is utilized to find the stop training criterion to get a smaller network size.

The rest of this paper is organized as follows. In Section 2 we describe the development of the proposed method. In Section 3 the examples and results are discussed. In Section 4 some conclusions are offered.

2. Methodology

2.1. Center and width of hidden kernel nodes

The theoretical basis of the RBF approach lies in the field of the interpolation of multivariate functions. In a Gaussian function, the closer to the center, the higher the emendation weights we will get. The center candidates make up the training data set. The center of the hidden layer neuron is determined as the one that has the smallest MSE in the training data

$$\mathrm{MSE} \equiv \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|^2, \tag{3}$$

where N is the number of data; y_i is desired output of the neural network; and $\hat{y}_i$ is actual output of the neural network.

It should be noted that different widths of hidden layer neurons cover different ranges, that is to say, different revised ranges. In previous RBF studies it has been most common to fix a value in the formulae to represent the width of the neurons, and this width decreases with the addition of hidden layer neurons. This is the static way to decide on the width of hidden layer neurons. In the proposed method, the nearest neighbor concept is used to determine the width. The width of new additional neurons is the smallest distance between the old neuron center and the new neuron center as in (4).

$$\rho^{new} = \min \left\| centre^{new} - centre^{old} \right\|, \tag{4}$$

where ρ^{new} and $centre^{new}$ are the width and center of the new additional neuron; $centre^{old}$ is center of the determined neurons.

The nearest neighbor (***nn***) is a kind of the simple classifier. The concept is to generate a prototype from the training data. Testing data are used to calculate the distance from the prototype. The testing data belongs to the prototype class which has the smallest distance.

The data around the center of a hidden layer neuron will receive a higher Gaussian function weight. The center of the hidden layer neuron is determined based on the smallest MSE in the network. From the view of the ***nn*** classifier, if the center of the hidden layer neuron is the prototype, then the data around the center of the hidden layer neuron belongs to the same class as the center. This group needs more correction as does the center of the hidden layer neuron. Hence, in this study, we apply this dynamic neuron width determination method after first neuron. The first neuron width is the maximum distance between the center and the training data set.

2.2. Orthogonalization

The Gram–Schmidt algorithm is a well-known standard numerical method that can be employed for orthogonalization of a basis function (Huang et al., 2008). However, here we replace the Gram–Schmidt algorithm with the QR algorithm. The QR algorithm can be used to solve a system like (5)

$$\boldsymbol{AX} = \boldsymbol{b}. \tag{5}$$

The associative law for matrix multiplication implies that in the factored system $(\boldsymbol{QR})\boldsymbol{X} = \boldsymbol{b}, \boldsymbol{Q}$ is an orthogonal matrix; and $\boldsymbol{R}$ is an upper triangular matrix that solves system (5). The QR algorithm is chosen for use because of its greater stability and improved calculation speed. To add a new neuron to the RBF neural network, it is simply necessary to add a new column to $\boldsymbol{A}$ by solving the OLS while the rest of $\boldsymbol{A}$ remains the same (Broomhead & Lowe, 1998; Moody & Darken, 1989). With the Gram–Schmidt algorithm on the other hand, it is necessary to re-solve (5) as a new problem. Since it is not necessary to do this with the QR algorithm we can thus save on calculation time (Parlett, 2000).

2.3. Improving generalization performance

The gradual increase in neurons in the neural network, should lead to a gradual drop of errors in the training data, and a gradual increase in errors in the testing data. The reason is that there are too many RBF hidden layer neurons, the so-called overfitting phenomenon. It can thus be seen that although at this moment the RBF neural network may have a better training performance it has poorer generalization performance. This is why we apply cross-validation of the input data for the RBF neural network includes the training data set, validation data set and testing data set. The training data set is used to construct the RBF neural network, the validation data set is used to stop the RBF neural network so as to get higher generalization performance (by stopping early), and the testing data set is used to examine the generalization performance.

The usage of a dynamic early stop to the training process can drastically improve the generalization performance of the neural network training process over that obtained using the number of hidden neurons as the upper limit and the admissible error as the lower limit. When there is no concept of neural network precision, the parameters may possibly need to be selected by trial and error. Early stopping can be regarded as a training condition to let the network stop the training process dynamically.

The early stop is applied by first allowing the number of hidden layer neuron to increase gradually. When the MSE of the validation data increases by two neurons in succession, that is to say the validation MSE for $p+1$ and $p+2$ neurons proves greater than for p neurons, then we define $p+2$ neurons as where the early stop takes place. The RBF neural network training process stops and the RBF neural network is set back to the condition of p neurons with better validation error.

If an early stop is regarded as one of the training criterion of the neural network, the network training can be stopped when generalization performance improves. The generalization performance is the most important property of the neural network. In addition to the early stop we also apply a statistical linear regression technique to improve the generalization performance.

The statistical linear regression technique is used to analyze the input and output variables from which we build a model for input output mapping. The input can consist of one variable (called simple regression) or multiple variables (called multivariable regression). The input output mapping is linear, so is called linear regression. In an RBF neural network, the input–output mapping can be both linear and nonlinear. Consequently the concept is the same.

Robust regression is a method for improving the robustness of the predicted regression results. In other words, the results of regression should be less susceptible to influence by outliers. In this study we use the bootstrap robust regression technique. The bootstrap method can be used to find the confidence interval. The concept comes from probability theory where training materials are divided repeatedly into ‘used data sets’ and ‘idle data sets’.