Information Sciences 381 (2017) 250–269

Contents lists available at ScienceDirect

Information Sciences

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

Convergence analyses on sparse feedforward neural networks via group lasso regularization[☆]

Jian Wang[a,d,*], Qingling Cai[b], Qingquan Chang[c], Jacek M. Zurada[d,e]

[a] *College of Science, China University of Petroleum, Qingdao 266580, China*
[b] *School of Engineering, Sun Yat-sen University, Guangzhou 510275, China*
[c] *School of Mathematics and Statistics, Lanzhou University, Lanzhou 730000, China*
[d] *Department of Electrical and Computer Engineering, University of Louisville, Louisville, KY 40292, USA*
[e] *Information Technology Institute, University of Social Sciences, Łódź 90-113, Poland*

ARTICLE INFO

Article history:
Received 21 February 2016
Revised 14 October 2016
Accepted 26 November 2016
Available online 28 November 2016

Keywords:
Clarke gradient
Convergence
Feedforward neural networks
Group Lasso
Non-differentiability

ABSTRACT

In this paper, a new variant of feedforward neural networks has been proposed for a class of nonsmooth optimization problems. The penalty term of the presented neural networks stems from the Group Lasso method which selects hidden variables in a grouped manner. To deal with the non-differentiability of the original penalty term ($l_1 - l_2$ norm) and avoid oscillations, smoothing techniques have been used to approximate the objective function. It is assumed that the training samples are supplied to the networks in a specific incremental way during training, that is, in each cycle samples are supplied in a fixed order. Then, under suitable assumptions on learning rate, penalization coefficients and smoothing parameters, the weak and strong convergence of the training process for the smoothing neural networks have been proved. The convergence analysis shows that the gradient of the smoothing error function approaches zero and the weight sequence converges to a fixed point, respectively. We demonstrate how the smoothing approximation parameter can be updated in the training procedure so as to guarantee the convergence of the procedure to a Clarke stationary point of the original optimization problem. In addition, we have proved that the original nonsmoothing algorithm with $l_1 - l_2$ norm penalty converges consistently to the same optimum solution with the corresponding smoothed algorithm. Numerical simulations demonstrate the convergence and effectiveness of the proposed training algorithm.

1. Introduction

Artificial neural networks have been widely used in various applications, such as pattern recognition, machine learning, data mining and signal processing [22,25,35]. The feedforward networks are one of the most popular architectures for their strong structural flexibility, good representation ability and compatibility with different training algorithms.

[☆] This workwas supported in part by the National Natural Science Foundation of China (No. 61305075), the China Postdoctoral Science Foundation (No. 2012M520624), the Natural Science Foundation of Shandong Province (No. ZR2013FQ004), the Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20130133120014) and the Fundamental Research Funds for the Central Universities (No. 13CX05016A, 14CX05042A, 15CX02079A, 15CX05053A, 15CX08011A).

[*] Corresponding author.
E-mail addresses: wangjiannl@upc.edu.cn (J. Wang), jacek.zurada@louisville.edu (J.M. Zurada).

http://dx.doi.org/10.1016/j.ins.2016.11.020
0020-0255/© 2016 Elsevier Inc. All rights reserved.

A reasonable architecture is one of the key aspects to guarantee better generalization of the trained neural networks [2]. Generally, the number of neurons in the input and output layers is fixed and represents the attributes and the target values of the dataset, respectively. Whereas the number of neurons in the hidden layer (or layers) depends on the complexity of the problem to be modeled, it is essential how to optimize the number of hidden neurons.

Typically, there are two approaches to determine the number of hidden neurons. One is the growing method, which starts with a small initial network and then adds hidden neurons stepwise during training [33,37,55].

The other is a pruning way, which starts with a large initial network and then removes the unnecessary neurons or weights [1,27,36,38,43,53]. Researchers have developed a number of pruning algorithms to optimize network architectures. An interesting comparative study for pruning algorithms of neural networks has been presented in [2]. Based on the elimination techniques, pruning methods could be categorized as penalty term methods, cross validation methods [28], magnitude base methods [19], evolutionary pruning methods [45], mutual information (MI) [16], significance based pruning methods and sensitivity analysis (SA) method [1,3]. This paper focuses on pruning technique by using a novel penalty term for back-propagation (BP) neural networks.

For standard networks, the error function is defined as the sum of the squared errors

$$E = \frac{1}{2}\sum_j \|Output^j - Target^j\|^2, \tag{1}$$

where j represents the jth sample of the dataset. When a network is to be pruned, it is common to add a penalty term to the error function during training

$$E = \frac{1}{2}\sum_j \|Output^j - Target^j\|^2 + \lambda\phi(\mathbf{w}). \tag{2}$$

The penalty term is to suppress the unnecessary connections between neurons. The parameter $\lambda > 0$ is the regularization coefficient, which balances the relative importance of the penalty term and the pure error expression.

There are three typical regularization forms for feedforward networks: *weight decay* [9,24,39,43], *weight elimination* [29,32,38,44] and *approximate smoother* [15,34].

When the backpropagation method is employed to train the network, uniform weight decay in [24] has a disadvantage that large weights are decaying at the same rate as small weights [20]. To remedy this problem, the weight elimination method has been suggested in [44]. Unfortunately, it does not distinguish between the large and very large weights [38]. The approximate smoother [48] appears to be more accurate than weight decay or weight elimination from the complexity point of view. It is designed for a multilayer perceptron with a single hidden layer and a single output neuron. However, it is more complicated than weight decay or weight elimination and has additional computational cost [21].

We notice that the above penalty methods discourage specific connections among neurons. To improve the interpretability and sparsity of neural networks, we will borrow the idea of Group Lasso to optimize the network architecture in this paper.

Lasso, the "least absolute shrinkage and selection operator" was first proposed for linear regression problem as a new technique that reduces some coefficients and sets others to zero [42]. It has been popular for simultaneous estimation and variable selection. However, lasso often results in selecting more factors than necessary and the solution depends on how the factors are represented. Then, a new version of the lasso, the adaptive lasso, was proposed in [56] by employing adaptive weights with l_1 penalty term. In addition, it enjoys the oracle properties.

An extension of lasso known as Group Lasso has been developed in [54] which selects the final models on the solution paths and encourages sparsity at group level. The penalty function ($l_1 - l_2$ norm) is intermediate between the l_1 penalty in lasso and the l_2 penalty in ridge regression (weight decay). In addition, a more general form, sparse Group Lasso, has been investigated in [17] which blends the lasso with the Group Lasso. Its main advantage is that it yields the sparse solutions at both the group and individual feature levels.

A novel idea of this paper is to replace the common penalty terms in [24] with a Group Lasso counterpart for BP networks. We expect that this can enhance some of the desirable properties of Group Lasso and improve the pruning performance with better generalization.

For a multi-layer network, we denote by $\mathbf{W}_i$ the weight matrix connecting the ith and $(i+1)$th layers. We suggest using the following expression

$$\phi(\mathbf{w}) = \sum_{i=1}^{n_l-1}\sum_{l=1}^{cl_i}\left\|\mathbf{w}_l^{(i)}\right\|, \tag{3}$$

where n_l is the number of layers, $\mathbf{w}_l^{(i)}$ is the lth column vector of matrix $\mathbf{W}_i$, cl_i is the number of column vectors in matrix $\mathbf{W}_i$. Thus the function of the optimization problem for pruning the hidden layer can be formulated as follows

$$E = \frac{1}{2}\sum_j \left\|Output^j - Target^j\right\|^2 + \lambda\sum_{i=1}^{n_l-1}\sum_{l=1}^{cl_i}\left\|\mathbf{w}_l^{(i)}\right\|, \tag{4}$$