Advanced Engineering Forum *Online: 2012-09-26*
ISSN: 2234-991X, Vols. 6-7, pp 1214-1219
doi:10.4028/www.scientific.net/AEF.6-7.1214

Statistical Sensitivity Analysis for Training Feedforword Neural Networks by B-splines Weight Functions

Daiyuan Zhang[1, 2, 3]

[1]College of Computer, Nanjing University of Posts and Telecommunications, Nanjing, China

[2]Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing, China

[3]Institute of Computer Technology, Nanjing University of Posts and Telecommunications, Nanjing, China

E-mail: dyzhang@njupt.edu.cn, zhangdaiyuan2011@sina.com

Keywords: neural network, weight function, B-splines function, statitical sensitivity.

Abstract. Based on the new architecture and algorithm of training neural networks by B-splines weight functions, the statistical sensitivity analysis for neural networks using B-splines weight functions is discussed in this paper. The sensitivity formula of B-splines weight function neural networks is derived. Finally, the correctness of theoretical sensitivity formula is verified by simulation.

Introduction

Sensitivity refers to how a system output is influenced by its input perturbations. Sensitivity analysis for neural networks can be dated back to the 1960s. Many scholars have studied sensitivity or the sensitivity of neural networks [1, 2]. In 1962, Hoff used an *n*-dimensional hypersphere to model Adaline for sensitivity analysis. Winter (1989) was the first one to derive an analytical expression for the probability of error in Madaline caused by weight perturbations. Stevenson continued Winter's work, established the sensitivity of Madaline to weight error, defined the mathematical model of sensitivity as hypersphere and analyzed the sensitivity of ADALINE. Piché [3] used a statistical approach to relate the output error to the change of weights for an ensemble of Madalines, with several activation functions such as linear, sigmoid, and threshold. On the related literature, other sensitivity function has been defined, such as output sensitivity, trajectory sensitivity, function sensitivity, etc.

By early algorithms (such as backpropagation (BP) or radial basis function (RBF) algorithm), the trained weights were constant data (constant weights). For the constant weights, let us assume that the connection weight vector changes from W^* to $W=W^*+\Delta W$, where W^* is the matrix of constant weights for the global minima of neural networks, and ΔW indicates the weight perturbations. Under the assumption of statistical weight perturbations, the statistical sensitivity can be defined as follows [2].

$$S(W^*) := \lim_{\sigma_{X_0^*} \to 0} \frac{[\mathrm{var}[\Delta Y_{L-1}]]^{1/2}}{\sigma_{W^*}} \tag{1}$$

It is well known that artificial neural networks have been widely used in many fields [4]. BP algorithm is trained by the steepest descent-like algorithms and can not find global minima in many applications. Many techniques have been introduced to improve the performance of the steepest descent-like algorithms. Those studies mainly focus on: improving the learning speed [5, 6], changing the network's architecture, growing and pruning algorithm [7, 8], optimizing the initialized weights or some other values [9], convergence of online training procedure [10], using more additional parameters, deterministic convergence, online gradient method [11], tuning network's parameters, neural network's algorithms using genetic or evolutionary methods [12], hybrid neural networks and so on. Unfortunately, the drawbacks can not be overcome completely arising from the steepest descent-like algorithms, such as the local minima, the slow convergence speed and the limited scale of problems. Therefore, the statistical sensitivity defined in (1) can hardly be used to find correct results.

In recent years, a new class of algorithms for training feedforward neural networks has been proposed by Zhang [13], who changes the weights from constant data to the cubic spline functions (weight functions), and the arguments of the cubic spline functions are input patterns.

The new algorithms proposed by [13] not only simplify the architecture of neural networks, but also overcome the drawbacks by using early algorithms, such as local minima, slow convergence and difficult to obtain the global optimal point.

Distinct form [13], a new algorithm for training neural networks using B-spline weight functions are introduced in this paper and the mathematic expression for statistical sensitivity is also obtained, in which the definition of statistical sensitivity is different from (1). Finally, the simulation examples show that the correctness of theoretical sensitivity formula derived from this paper.

Fundamentals of training neural networks by B-splines Weight Functions

Fig. 1 shows a model of B-splines function neural network, where $b_i(x_i)$ denotes B-splines weight function.

Suppose there are $N+2$ patterns, and $w_i(x_i)$ is the theoretical weight function which represents the connection with the *i*-th $(i=1,2\cdots m)$ input node x_i $(i=1,2\cdots m)$, i.e. the *i*-th component of *m* dimensional input vectors,

$$\boldsymbol{x_i} = (x_{i0}\ \ x_{i1}\ \cdots\ x_{i(N+1)}) \tag{2}$$

$$\boldsymbol{z} = (z_0\ \ z_1\ \cdots\ z_{N+1}) \tag{3}$$

$$\boldsymbol{z_i} = \left(w_i(x_{i0})\quad w_i(x_{i1})\quad \cdots\quad w_i\left(x_{i(N+1)}\right)\right) = (\eta_i z_0\quad \eta_i z_1\quad \cdots\quad \eta_i z_{N+1}) \tag{4}$$

where $\sum_{i=1}^{m}\eta_i = 1$, the interpolate nodes can be expressed as

$$\boldsymbol{Ip_i} = \left\{Ip_{i0}, Ip_{i1}, \cdots Ip_{i(N+1)}\right\} = \left\{(x_{i0}, \eta_i z_0), (x_{i1}, \eta_i z_1)\cdots (x_{i(N+1)}, \eta_i z_{N+1})\right\} \tag{5}$$

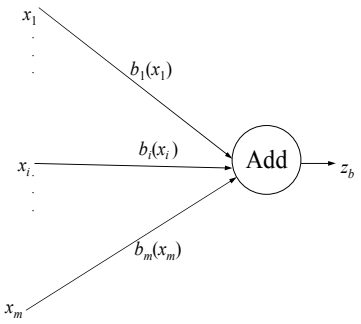

Fig. 1 The architecture of neural network using B-splines function

The approximate weight function $b_i(x_i)$ and the theoretic weight function $w_i(x_i)$ have the same value on the interpolation points, but there are some errors out of the interpolation points.

Let $\cdots x_{-k} < \cdots < x_0 < \cdots < x_N < \cdots < x_{N+k} \cdots$, we have

$$N_{i,k}(x) = (x_{i+k+1} - x_i)[x_i, x_{i+1}, \cdots x_{i+k+1}](t-x)_+^k = (x_{i+k+1} - x_i)\sum_{j=i}^{i+k+1}\frac{(x_j - x)_+^k}{\Pi'_{i,k+1}(x_j)} \tag{6}$$

where $N_{i,k}(x)$ is called the *i*-th B-splines function of degree *k*, and

$$\Pi_{i,k+1}(x) = (x - x_i)(x - x_{i+1})\cdots(x - x_{i+k+1}) \tag{7}$$

Any spline function $b(x)$ can be expressed as

$$b(x) = \sum_{i=-k}^{N-1} c_i N_{i,k}(x) \tag{8}$$

where c_i is a constant.

For the interval $[a,b]$, the spline space is $N+k$ dimensional, and the function $w(x)$ has $N+k$ interpolation nodes, let the interpolation points be

$$\xi_{-k} < \xi_{-k+1} < \cdots \xi_{N-1} \tag{9}$$

The *k*-th spline interpolation satisfies the following conditions

$$b(\xi_i) = y_i,\ i = -k, -k+1, \cdots N-1 \tag{10}$$

Where y_i is the output value, and $b(x)$ satisfied (10) is called the *k*-th spline interpolation function. If we take $x_0, x_1, \cdots, x_N$ as spline interpolation nodes, from (10) and (8) we have

$$\sum_{i=-k}^{N-1} c_i N_{i,k}(\xi_i) = y_i\ (i = -k, -k+1, \cdots N-1) \tag{11}$$

We can solve the linear system (11) for finding the coefficient c_i.

Statistical Sensitivity Analysis of B-splines Weight Function Neural Networks

Different from (1), the definition of statistical sensitivity can be described as:

$$S\left(X_0^*\right) := \lim_{\sigma_{X_0^*} \to 0} \frac{\left[\mathrm{var}\left[\Delta Y_{L-1}\right]\right]^{1/2}}{\sigma_{X_0^*}} \tag{12}$$

Where $\sigma_{X_0^*}$ is standard deviation of the disturbances on input patterns, $S\left(X_0^*\right)$ represent the value of sensitivity.

When noise Δx is embedded, the theoretic error of B-splines weight function neural networks can be expressed as

$$\begin{aligned} &\|w(x+\Delta x) - w(x)\| = \|w(x+\Delta x) - b(x+\Delta x) + b(x+\Delta x) - b(x) + b(x) - w(z)\| \\ &\le \|w(x+\Delta x) - b(x+\Delta x)\| + \|b(x+\Delta x) - b(x)\| + \|w(z) - b(x)\| \end{aligned} \tag{13}$$

Where $\|\ldots\|$ represents some kinds of norms. Generally speaking, the theoretical noise errors of neural networks contain model error and approximation noise error.

Firstly, we will analyze model error $\|w(x) - b(x)\|$. Many norms can be adapted to measure errors. Here we use Chebyshev norm. For continuous function, Chebyshev norm is the maximum of absolute value. That is

$$\|w(x) - b(x)\|_\infty = \max_{x \in [x_0, x_{N+1}]} |w(x) - b(x)| \tag{14}$$

If $w(x)$ and $b(x)$ in the interval $[x_0, x_{N+1}]$ are continuous functions, then $w(x)$ and $b(x)$ in the interval $[x_0, x_{N+1}]$ can get the maximum values, we have

$$\|w(x) - b(x)\|_\infty = \max |w(x) - b(x)| = \max_p |w_p(x) - b_p(x)| \le \max_p \left| \|w_p(x)\|_\infty - c^* \right| = \left| \|w(\mathrm{x})\|_\infty - \overline{c}^* \right| \tag{15}$$

where *p* is the index of subinterval $\left[x_p, x_{p+1}\right]$, and $p \in \{0,1,2,\cdots,N\}$, $\overline{c}^* = \min\{c_{-k}, \cdots, c_{N-1}\}$.

Suppose the input without perturbation is $x \in \left[x_p, x_{p+1}\right]$, the input perturbation is Δx and $(x+\Delta x) \in \left(x_q, x_{q+1}\right)$, then the approximation error will be

$$b(x+\Delta x) - b(x) = \sum_{i=-k}^{N-1} c_i N_{i,k}(x+\Delta x) - \sum_{i=-k}^{N-1} c_i N_{i,k}(x) = \sum_{i=-k}^{N-1} c_i \left[N_{i,k}(x+\Delta x) - N_{i,k}(x)\right] \tag{16}$$

From (16) and (6), we can get

$$b(x+\Delta x) - b(x) = -\Delta x k \sum_{i=-k}^{N-1} \sum_{j=i}^{i+k+1} c_i h_i \frac{\left(x_j - x\right)_+^{k-1}}{w'_{i,k}\left(x_j\right)} \tag{17}$$

where $h_i = x_{i+k+1} - x_i$.

Let the dimension of input and output be *m* and *n* respectively, $z_j = \sum_{i=1}^{m} b_{ji}(x_i)$ and $z_j^* = \sum_{i=1}^{m} w_{ji}(x_i)$ represent approximate output and target output respectively, the output perturbation is

$$\Delta z_j = z_j^* - z_j = \sum_{i=1}^{m} \left\{ \left[w_{ji}(x_i + \Delta x_i) - b_{ji}(x_i + \Delta x_i) \right] - \left[w_{ji}(x_i) - b_{ji}(x_i) \right] + \left[b_{ji}(x_i + \Delta x_i) - b_{ji}(x_i) \right] \right\}$$

$$\le \sum_{i=1}^{m} \left(\left\| w_{ji}(x_i + \Delta x_i) \right\|_\infty + \left| \overline{c}_{ji}^* \right| \right) - \sum_{i=1}^{m} \left(\left\| w_{ji}(x_i) \right\|_\infty + \left| \overline{c}_{ji}^* \right| \right) - k \left[\begin{array}{l} \sum_{i=1}^{m} \sum_{l=q-k}^{q} \sum_{r=l}^{l+k+1} \Delta x_i c_{(ji),l} h_{i,l} \dfrac{(x_r - x_i)_+^{k-1}}{w'_{l,k}(x_r)} \\ + \sum_{i=1}^{m} \sum_{l=p-k}^{p} \sum_{r=l}^{l+k+1} \Delta x_i c_{(ji),l} h_{i,l} \dfrac{(x_r - x_i)_+^{k-1}}{w'_{l,k}(x_r)} \end{array} \right] \tag{18}$$

If the function $w_{ji}(x_i)$ ($j = 1, 2, \cdots, n$) is continuous, then

$$\left\| w_{ji}(x_i) \right\|_\infty = \max \left| w_{ji}(x_i) \right| \tag{19}$$

Let

$$\left\| w_{ji}(x_i) \right\|_\infty + \left| \overline{c}_{ji}^* \right| = a_j(x_i), \; p_l(x_i) = \sum_{r=l}^{l+k+1} \frac{(x_r - x_i)^{k-1}}{w'_{l,k}(x_r)} \tag{20}$$

we have

$$\begin{aligned} \Delta z_j &= z_j^* - z_j = (\Delta x_1, \Delta x_2, \cdots, \Delta x_m) \left(a'_j(x_1), a'_j(x_2), \cdots, a'_j(x_m) \right)^T \\ &+ k(\Delta x_1, \Delta x_2, \cdots, \Delta x_m) \cdot \left(\sum_{l=p-k}^{p} h_l c_{(j1)l} p_l(x_1), \sum_{l=p-k}^{p} h_l c_{(j2)l} p_l(x_2), \cdots, \sum_{l=p-k}^{p} h_l c_{(jm)l} p_l(x_m) \right)^T \end{aligned} \tag{21}$$

Therefore, the sensitivity of B-splines weight function can be expressed by (12) in the following

$$S(X_0^*) := \lim_{\sigma_{X_0^*} \to 0} \frac{[\mathrm{var}[\Delta Y_{L-1}]]^{1/2}}{\sigma_{X_0^*}} = \sqrt{E\left(\boldsymbol{A}'(\boldsymbol{X}) - k \sum_{l=q-k}^{q} H_l .* C_l .* P_l(\boldsymbol{X}) - k \sum_{l=p-k}^{p} H_l .* C_l .* P_l(\boldsymbol{X}) \right)^2} \tag{22}$$

Simulations

To show the results proposed in this paper, an example is given below. The architecture of the network is 3-4, and the learning cure [13] is

$$\begin{cases} x_1 = t \\ x_2 = 2t + t^2 \\ x_3 = 2 + t \end{cases} \tag{23}$$

The output patterns are obtained by

$$\begin{cases} y_1 = \sin(x_1 + x_2 + x_3) \\ y_2 = \cos(x_1 + \pi x_2 + x_3) \\ y_3 = \exp(-(x_1 + x_2 + x_3)) \\ y_4 = -(x_1 + x_2 + x_3) \end{cases} \tag{24}$$

Then we can get B-splines weight functions by the training patterns. The statistical sensitivity of B-splines weight function neural network can be calculated by (22). For the trained neural network, suppose the output without disturbance is Y^*, and the disturbed output is Y, then relative error can be got by the formula as follows.

$$\left| \frac{Y^* - Y}{Y} \right| \tag{25}$$

When the input perturbation is very small, theoretical noise sensitivity matches approximate noise sensitivity, see Fig. 2 Fig. 3 illustrates that the output relative error is accordingly very small (close to zero), so we can conclude that the network has generation ability. As can be seen from Fig. 2 and Fig. 3, when the input perturbations increase to some extent, the sensitivity changes quite so obvious, i.e. the sensitivity may reduce, may increase, or increase and then decrease. But the output relative error increases rapidly at the same time. The network doesn't remain stable, very sensitive to the input perturbations and the output deviates from the target values.

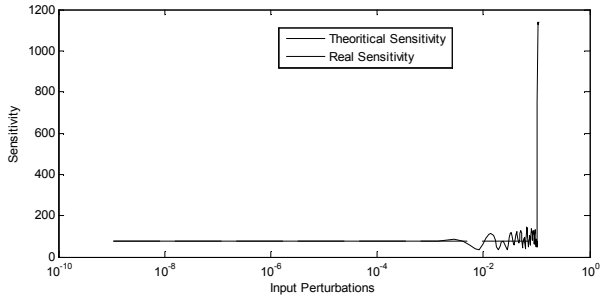

Fig. 2 Sensitivity of B-splines weight functions

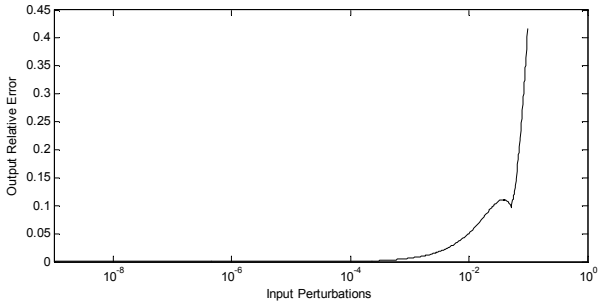

Fig. 3 Relative errors

Conclusions

Although some results of statistical sensitivity analysis have been achieved using early algorithms, further analysis of the method proposed in this paper is necessary for improved insight into its effectiveness. The sensitivity formula of B-splines weight function neural networks is derived and the correctness of theoretical sensitivity formula is verified by simulations. It can be seen that the theoretical value of sensitivity is determined by several factors. The sensitivity depends not only on the weight functions, but also on the input patterns, which could be determined by measuring the sensitivity of the network.

Acknowledgment

This work was supported by the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (yx002001).

I thank Y. Dong, my graduate student, for her simulation examples given in this paper.

References

[1] M. Stevenson, R.Winter, and B. Widrow. Sensitivity of feedforward neural networks to weight errors. IEEE Trans. Neural Netw., 1990,1(1): 71-80.

[2] J.Y Choi and C. Choi. Sensitivity analysis of multilayer perception with differentiable activation functions. IEEE Trans. Neural Netw., 1992,3(1): 101-107.

[3] S. W. Piché. The selection of weight accuracies for Madalines. IEEE Trans. Neural Netw., 1995,6(2): 432-445.

[4] Martin T.Hagan, Howard B.Demuth, Mark Beale. Neural Network Design. Beijing: China Machine Press, 2002.

[5] S. Ergezinger, E. Tomsen. An accelerated learning algorithm for multilayer perceptrons: optimization layer by layer. IEEE Trans. Neural Netw., 1995, 6(1): 31-42.

[6] N. Ampazis, S. J. Perantonis. Two highly efficient second-order algorithms for training feedforward networks. IEEE Trans. Neural Netw., 2002, 13 (5): 1064-1073.

[7] G. B. Huang, P. Saratchandran, N. Sundararajan. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans. Neural Netw., 2005, 16(1): 57-67.

[8] M. Bortman, M .Aladjem. A growing and pruning method for radial basis function networks. IEEE Trans. Neural Netw., 2009, 20(6): 1039-1045.

[9] L. Behera, S. Kumar, A. Patnaik. On adaptive learning rate that guarantees convergence in feedforward networks. IEEE Trans. Neural Netw., 2006, 17(5): 1116-1125.

[10] Z. B. Xu, R. Zhang, W. F. Jing. When does online BP training converge?. IEEE Trans. Neural Netw., 2009, 20(10): 1529-1539.

[11] W. Wu, G. R. Feng, Z.X. Li, Y. S. Xu. Deterministic convergence of an online gradient method for BP neural networks. IEEE Trans. Neural Netw., 2005, 16(5): 533-540.

[12] A. Khashman. A modified backpropagation learning algorithm with added emotional coefficients. IEEE Trans. Neural Netw., 2008, 19(11): 1896-1909.

[13] D. Y. Zhang. New Theories and Methods on Neural Networks. Beijing: Tsinghua University Press, 2006 (in Chinese).