Available online at www.sciencedirect.com

ScienceDirect

Neurocomputing 71 (2008) 1515–1526

NEUROCOMPUTING

www.elsevier.com/locate/neucom

A definition of partial derivative of random functions and its application to RBFNN sensitivity analysis

Xi-Zhao Wang[a], Chun-Guo Li[a,*], Daniel So Yeung[b], ShiJi Song[c], HuiMin Feng[a]

[a]*Department of Mathematics and Computer Science, Hebei University, Baoding, Hebei, China*
[b]*Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong*
[c]*Department of Automatization, Tsinghua University, Beijing, China*

Received 21 January 2006; received in revised form 24 October 2006; accepted 8 May 2007
Available online 7 June 2007

Abstract

Considering the inputs of a feed-forward neural network as random variables, this paper proposes a definition of partial derivative of a function with respect to a random variable in the probability measure space. The mathematical expectation of the mean square or absolute value of the partial derivative is regarded as a type of measure of the network's sensitivity, which extends Zurada's sensitivity definition of networks in Zurada et al, [Perturbation method for deleting redundant inputs of perceptron networks, Neurocomputing 14 (1997) 177–193] from the certain environment to the stochastic environment. Furthermore, for the purpose of network's redundant feature deletion or feature selection, the new sensitivity measure is applied to the sensitivity analysis of Radial Basis Function Neural Networks (RBFNNs). The feasibility and the effectiveness of the sensitivity approach to redundant feature deletion are illustrated.

Keywords: Partial derivative of a random variable; Probability measure space; Sensitivity measure; Mathematical expectation; Feature selection; RBFNN

1. Introduction

Radial Basis Function Neural Networks (RBFNNs) are powerful computational tools for regression and classification. Based on the training data sets, RBFNNs form a special structure, which employs the high dimension of the hidden layer and the nonlinear hidden neurons, to simulate the training data sets within a required accuracy. This special structure usually leads to a problem, that is, the hidden layers sometimes are too high to be studied. Therefore, it is necessary to introduce the simplification of the networks.

A number of researchers have proposed many methodologies to solve the network simplification problem, such as improving the training algorithm [1,2], sensitivity-based feature selection [3,4] and sensitivity analysis of the centers of the hidden neurons [5]. These methodologies aim to simplify the network without depressing its performance. Among these methodologies, the sensitivity-based network simplification is one of the most effective and most promising techniques.

Sensitivity analysis of features is a fundamental issue in neural network design. It can be used as the means of feature selection, simplifying the neural network and improving the generalization performance. A number of useful methodologies have been put forward to investigate the sensitivity of the networks [6, 10–12]. For a Multilayer Feed-Forward Neural Network (MFNN), one of the most popular techniques in network simplification is to delete redundant features by using the derivative-based sensitivity measure which is defined in [6] by Jacobian matrix. It is assumed in [6] that the network performs a nonlinear, differentiable mapping. To apply the sensitivity measure to delete the redundant inputs, Zurada et al. in [6] suggested several ways of averaging the set of training patterns. Zurada's methods to delete redundant inputs can also be used to the RBFNN since RBFNNs are one kind of feed-forward neural network with a single hidden layer and perform the nonlinear, differentiable mapping.

*Corresponding author.
E-mail address: licg@mail.hbu.cn (C.-G. Li).

0925-2312/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.05.005

Considering the inputs of a feed-forward neural network as random variables, this paper proposes a definition of partial derivative of a random function with respect to a random variable in the probability measure space. Noting that in [6] the sensitivity measure is defined as the partial derivative in common sense, this paper considers the mathematical expectation of the partial derivative as a type of measure of the network's sensitivity, which extends Zurada's sensitivity definition of networks in [6] from the certain environment to the stochastic environment. Furthermore, for the purpose of network's redundant feature deletion or feature selection, the new sensitivity measure is applied to the sensitivity analysis of RBFNNs. The feasibility and the effectiveness of the approach to redundant feature deletion are illustrated.

2. Definition of derivative with respect to a random variable

Let $f(x_1, x_2, \ldots, x_n)$ be a given real differentiable function with n real variables. We use the function f to denote the output of a feed-forward neural network and $x_1, x_2, \ldots, x_n$ to denote the network inputs (features). As assumed in [6], the network performs a nonlinear and differentiable mapping. According to Zurada et al. [6] where the network output is multi-values, the network sensitivity at an input point $(a_1, a_2, \ldots, a_n)$ is defined as the vector

$$\left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right)\Bigg|_{(a_1, a_2, \ldots, a_n)}. \tag{2.1}$$

Then, Zurada et al. suggested in [6] that the network sensitivity on a training set is resulted from (2.1). Over a given training set

$$D = \left\{ \left(a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)}\right)\Big|_{j=1,2,\ldots,N} \right\},$$

one type of network sensitivity, i.e., the mean square average sensitivity, is defined as $(s_1, s_2, \ldots, s_n)$

$$s_i = \sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(\frac{\partial f}{\partial x_i}\left(a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)}\right)\right)^2}, \tag{2.2}$$

where N is the number of patterns in the given training set.

This paper is aiming at generalizing the network sensitivity (2.2) from the case of real variables $x_1, x_2, \ldots, x_n$ to the case of random variables $\xi_1, \xi_2, \ldots, \xi_n$. In order to accomplish this extension, first of all, the definition of partial derivative in the probability space is given.

Some definitions of a random process with respect to time variable t can be found in Refs. [e.g. 13–16]. For example, suppose one defined random process $\{X(t), t \in T\}$ has the second moment. The random process is *mean square differential* at t if $(X(t+h) - X(t))/h$ converges to $X'(t)$ in mean square for given $t \in T$. $X'(t)$ is defined as the *mean square derivative* of $X(t)$. $X'(t)$ is still one random process having finite second moment, i.e. $E[X'(t)]^2 < \infty$.

Note that this derivative definition was defined with respect to the time item, not to the random variable. To generalize the sensitivity definition referred in [6], we need to consider the derivative of the function of random variable, with respect to its random variable. Motivated by the sensitivity improvement, the new derivative definition is given below with respect to its random variable.

Definition 2.1. [7] Let $\xi, \xi_1, \xi_2, \ldots, \xi_n$, be a series of random variables defined on a probability space (Ω, A, P). $\{\xi_n\}$ is said to *converge in probability* to ξ if and only if the following limit:

$$P(\{|\xi_n - \xi| \geqslant \varepsilon\}) \to 0 (n \to \infty) \tag{2.3}$$

holds true for $\forall \varepsilon > 0$.

We use $\xi_n \xrightarrow{P} \xi (n \to \infty)$ to denote that $\{\xi_n\}$ converges in probability to ξ.

Definition 2.2. Let ξ be a random variable defined on a probability space (Ω, A, P) and $f(x)$ be a real differentiable function. Replacing the real variable x with random variable ξ inside f, we can construct a series of random variables

$$\tau_n = \frac{f(\xi + \Delta_n) - f(\xi)}{\Delta_n} \tag{2.4}$$

for any fixed real sequence $\{\Delta_n | \Delta_n \neq 0, \ n = 1, 2, \ldots\}$ converging to 0. If there exists a random variable τ such that the sequence $\{\tau_n\}$ converges in probability to τ, then we call the random variable τ as the *derivative* of $f(\xi)$ with respect to ξ, denoted by $f'(\xi)$.

Theorem 2.1. *Let ξ be a random variable and $f(x)$ be a real differentiable function. Then the following equality holds*:

$$f'(\xi) = \frac{\mathrm{d}f}{\mathrm{d}x}(\xi), \tag{2.5}$$

where the left is defined as in Definition 2.2 and the right means that the real variable x in the real function $\mathrm{d}f/\mathrm{d}x$ is replaced with the random variable ξ.

Proof. The correctness of Eq. (2.5) is equivalent to prove that

$$P\left\{\left|\frac{f(\xi + \Delta_n) - f(\xi)}{\Delta_n} - \frac{\mathrm{d}f}{\mathrm{d}x}(\xi)\right| \geqslant \varepsilon\right\} \to 0, \quad (n \to \infty) \tag{2.6}$$

holds true for $\forall \varepsilon > 0$. It is sufficient to verify

$$E\left(\frac{f(\xi + \Delta_n) - f(\xi)}{\Delta_n} - \frac{\mathrm{d}f}{\mathrm{d}x}(\xi)\right)^2 \to 0, \quad (n \to \infty)$$

is valid [7].

Noting that $\lim_{n \to \infty}(f(x + \Delta_n) - f(x))/\Delta_n = \mathrm{d}f/\mathrm{d}x$ implies that $|(f(x + \Delta_n) - f(x))/\Delta_n - \mathrm{d}f/\mathrm{d}x| < \varepsilon$ for $\forall \varepsilon > 0$, we obtain

$$\left(\frac{f(x + \Delta_n) - f(x)}{\Delta_n} - \frac{\mathrm{d}f}{\mathrm{d}x}\right)^2 < \varepsilon^2.$$

According to the Control and Convergence Theorem [9], the above formula can be followed by

$$E\left(\frac{f(\xi+\Delta_n)-f(\xi)}{\Delta_n}-\frac{\mathrm{d}f}{\mathrm{d}x}(\xi)\right)^2$$
$$=\int_{-\infty}^{\infty}\left(\frac{f(x+\Delta_n)-f(x)}{\Delta_n}-\frac{\mathrm{d}f}{\mathrm{d}x}\right)^2\varphi(x)\,\mathrm{d}x$$
$$<\varepsilon^2\int_{-\infty}^{\infty}\varphi(x)\,\mathrm{d}x=\varepsilon^2,$$

where $\varphi(x)$ is the density function of random variable ξ. Thus (2.6) is proved to be valid. The proof is completed.

Similar to the derivative of real-valued function, the derivative given in Definition 2.2 has the following simple properties.

Theorem 2.2. *Let ξ be a random variable, $f(x)$ and $g(x)$ be two real differentiable functions, and c_1, c_2 be two real constants. Then the following equalities hold*:

(i) $$(c_1f(\xi)+c_2g(\xi))'=c_1f'(\xi)+c_2g'(\xi), \tag{2.7}$$

(ii) $$(f(\xi)\cdot g(\xi))'=f'(\xi)\cdot g(\xi)+f(\xi)\cdot g'(\xi), \tag{2.8}$$

where the derivatives are given by Definition 2.1.

Theorem 2.2 can be easily proved based on Theorem 2.1 according to the properties of the derivative of real-valued function.

Definition 2.2 can be easily extended to the case of multiple variables.

Definition 2.3. Let $\xi_1,\xi_2,\ldots,\xi_n$ be n random variables defined on a probability space (Ω,A,P) and $f(x_1,x_2,\ldots,x_n)$ be a real differentiable function. Replacing the ith real variable x_i with random variable ξ_i inside f for each $i(1\leqslant i\leqslant n)$, we can construct a series of random variables

$$\tau_n=\frac{f(\xi_1,\ldots,\xi_i+\Delta_n,\ldots,\xi_n)-f(\xi_1,\ldots,\xi_i,\ldots,\xi_n)}{\Delta_n} \tag{2.9}$$

for any fixed real sequence $\{\Delta_n|\Delta_n\neq 0,\ n=1,2,\ldots\}$ converging to 0. If there exists a random variable τ such that the sequence $\{\tau_n\}$ converges in probability to τ, then we call the random variable τ as the *partial derivative* of $f(\xi_1,\xi_2,\ldots,\xi_n)$ with respect to ξ_i, denoted by $f'_{(i)}(\xi_1,\xi_2,\ldots,\xi_n)$, in short, $f'_{(i)}$.

Similar to Theorem 2.1, we have the following Theorem 2.3.

Theorem 2.3. *Let $\xi_1,\xi_2,\ldots,\xi_n$ be n random variables and $f(x_1,x_2,\ldots,x_n)$ be a real differentiable function. Then we have*:

$$f'_{(i)}(\xi_1,\xi_2,\ldots,\xi_n)=\frac{\delta f}{\delta x_i}(\xi_1,\xi_2,\ldots,\xi_n), \tag{2.10}$$

where the left is defined as Definition 2.3 *and the right means that the real variables $x_1,x_2,\ldots,x_n$ inside the real function $\delta f/\delta x_i$ are replaced with the random variables $\xi_1,\xi_2,\ldots,\xi_n$ respectively.*

Proof. Similar to the proof of Theorem 2.1.

3. Extending Zurada's sensitivity from real variables to random variables

Based on the definitions and theorems proposed in Section 2, we can generalize the sensitivity Definition 2.2 by giving the following Definition 3.1.

Definition 3.1. Consider a trained differential feedforward neural network in which the inputs are random variables $\xi_1,\xi_2,\ldots,\xi_n$. The mean square average sensitivity vector is defined as $(s_1,s_2,\ldots,s_n)$ where

$$s_i=\sqrt{E\left[(f'_{(i)})^2\right]}$$
$$=\sqrt{\int(f'_{(i)}(x_1,x_2,\ldots,x_n))^2\mathrm{d}F(x_1,x_2,\ldots,x_n)}, \tag{3.1}$$

where $F(x_1,x_2,\ldots,x_n)$ denotes the joint distribution of random variables $(\xi_1,\xi_2,\ldots,\xi_n)$.

It is noted that Eq. (3.1) can be estimated on a sample set (training set). According to parameter estimation theory in mathematical statistics [8], (2.2) can be the best estimation of (3.1) over the training set $D=\{(a_1^{(j)},a_2^{(j)},\ldots,a_n^{(j)})|_{j=1,2,\ldots,N}\}$ when the distribution of $(\xi_1,\xi_2,\ldots,\xi_n)$ is never considered. But generally, the computation of (3.1) will depend on the joint distribution of $(\xi_1,\xi_2,\ldots,\xi_n)$. In this sense, our definitions (3.1) indeed generalizes (2.2) given by Zurada et al. in [6].

For one certain differential feedforward neural network, the calculating formulae of Eq. (3.1) are expected to be explicitly derived when the joint distribution is known. For illustration, we just deduce the sensitivity calculating formulae for one trained RBFNN in case that the random variables are independently and normally distributed.

3.1. Sensitivity measure of RBFNN with one output

As a feed-forward network performing a nonlinear and real differentiable mapping, a trained RBFNN can be expressed as follows:

$$f(x_1,x_2,\ldots,x_n)=\Sigma_{j=1}^m w_j\left(\exp\left(\frac{\Sigma_{i=1}^n(x_i-u_{ji})^2}{-2v_j^2}\right)\right), \tag{3.2}$$

where n is the number of features (i.e. the number of dimensions of input vector), m the number of centers, $(u_{j1},u_{j2},\ldots,u_{jn})$ the jth center, v_j the spread of the jth center, and w_j the weight of the output layer, $j=1,2,\ldots,m$. The derivative of the network to the ith variable is as following:

$$\frac{\partial f}{\partial x_i}=\sum_{j=1}^m w_j\exp\left(\frac{\Sigma_{k=1}^n(x_k-u_{kj})^2}{-2v_j^2}\right)\left(\frac{x_i-u_{ij}}{-v_j^2}\right). \tag{3.3}$$

Consider all inputs of a RBFNN as random variables $\xi_1,\xi_2,\ldots,\xi_n$. According to Theorem 2.3, after replacing the real variables $x_1, x_2, \ldots, x_n$ in Eq. (3.3) with random variables $\xi_1, \xi_2, \ldots, \xi_n$, we can get the following partial derivative:

$$f'_{(i)} = \sum_{j=1}^{m} w_j \exp\left(\frac{\Sigma_{k=1}^{n}(\xi_k - u_{kj})^2}{-2v_j^2}\right)\left(\frac{\xi_i - u_{ij}}{-v_j^2}\right). \tag{3.4}$$

Theorem 3.1. *Suppose that random variables $\xi_1, \xi_2, \ldots, \xi_n$ are independently and normally distributed with mean $(\mu_1, \mu_2, \ldots, \mu_n)$ and variances $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$. Then the mean square average sensitivity defined by* (3.1) *has the following computational formula*:

$$E[(f'_{(i)})^2] = \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \left(\frac{\sigma_i^2}{a_i} + \frac{1}{v_j^2 v_t^2}\left(\frac{b_i}{a_i} - u_{ij}\right)\left(\frac{b_i}{a_i} - u_{it}\right)\right) \times \prod_{k=1}^{n}\left[\frac{v_j v_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{(c_k - b_k^2)/a_k}{v_j^2 v_t^2 \sigma_k^2}\right)\right)\right]$$
$$i = 1, 2, \ldots, n, \tag{3.5}$$

where

$$a_k = v_t^2\sigma_k^2 + v_j^2\sigma_k^2 + v_j^2 v_t^2,$$

$$b_k = v_t^2\sigma_k^2 u_{kj} + v_j^2\sigma_k^2 u_{kt} + v_j^2 v_t^2 \mu_k,$$

$$c_k = v_t^2\sigma_k^2 u_{kj}^2 + v_j^2\sigma_k^2 u_{kt}^2 + v_j^2 v_t^2 \mu_k^2.$$

The proof is placed in Appendix A.

Definition 3.1 shows that the new sensitivity measure given by Definition 3.1 has extended the sensitivity measure given by Zurada et al. in [6]. The extension is achieved by considering all real inputs as random variables. Theorem 3.1 shows that for RBFNNs the computational formula of the sensitivity magnitude depends only on the numerical characteristics such as means and variances of the inputs (which are random variables). Since the means and variances of a random variable can be easily estimated from a given training set, formula (3.5) is easy to implement. One key point required to note is that the formula (3.5) is data-driven. It means that we can compute the sensitivity of a feature for the RBFNN once the dataset is given explicitly. Another key point is that the sensitivity definition given by (3.1) is independent of the perturbation of variables. There will not exist such a case that a feature has sensitivity α for a perturbation but has sensitivity $\beta(\neq\alpha)$ for another perturbation.

In comparison with the sensitivity measure given in [6], we list the following differences and similarities.

(1) This paper investigates only one type of feed-forward neural networks, i.e., RBFNNs while paper [6] investigates the general feed-forward neural networks.
(2) The network outputs in this paper are reduced into one value for simplicity while the paper [6] handles the multi-valued output.
(3) All real inputs of the network in [6] are here replaced with random variables.
(4) The sensitivity measure is based on the joint distribution of random input-variables instead of the training data.
(5) When the distribution information of the random inputs is not considered, the sensitivity estimation over a training set will degenerate to the case given in [6].
(6) Both sensitivity measures can be used for redundant feature deletion or feature selection of a feed-forward neural network.
(7) The computational complexity of the sensitivity measure proposed in this paper is $O(n \times m^2)$ but the corresponding complexity in [6] is $O(N \times n^2 \times m)$.
(8) With different distributions of the inputs, the sensitivity formulae (3.5) will change. The calculating formulae would be derived correspondingly.

Unfortunately, for the above item (8), when the random variables are non-Gaussian distribution, the calculating formulae are usually very difficult to derive, or very complex. For example, when the random variables are uniform distribution, the calculating formula (3.1) is (B.1), which can be found in Appendix B (the formula also illustrates the changes go with different distribution). To deal with this situation, we can recur to the approximation of the sensitivity measure (3.1) on the training set.

Given the training dataset $D = \{(a_1^{(j)}, a_2^{(j)}, \ldots, a_n^{(j)})|_{j=1,2,\ldots,N}\}$ the sensitivities can be estimated on the training dataset, taking the distribution into consideration. The sensitivity formula of ith feature of the network (3.1) can be rewritten as

$$s_i^2 = E\left[(f'_{(i)})^2\right] = \int (f'_{(i)}(x_1, x_2, \ldots, x_n))^2 \,\mathrm{d}F(x_1, x_2, \ldots, x_n). \tag{3.6}$$

Assuming the random variables are *all independently and continuously distributed*, Eq. (3.6) can go a further step:

$$s_i^2 = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}(f'_{(i)}(x_1, x_2, \ldots, x_n))^2 \,\mathrm{d}F(x_1, x_2, \ldots, x_n) = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}(f'_{(i)}(x_1, x_2, \ldots, x_n))^2 \prod_{k=1}^{n}\varphi(x_k)\,\mathrm{d}x_1\,\mathrm{d}x_2\ldots\mathrm{d}x_n, \tag{3.7}$$

where $\varphi(x_i)$ $(1 \leqslant i \leqslant n)$ is the density function of the ith feature. This integration can be approximated by

$$s_i^2 = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}(f'_{(i)}(x_1, x_2, \ldots, x_n))^2 \prod_{k=1}^{n}\varphi(x_k)\,\mathrm{d}x_1\,\mathrm{d}x_2\ldots\mathrm{d}x_n = \sum_{j=1}^{N}(f'_{(i)}(x_1^j, x_2^j, \ldots, x_n^j))^2 \prod_{k=1}^{n}\varphi(x_k^j)\,\mathrm{d}\sigma_j, \tag{3.8}$$

where $(x_1^j, x_2^j, \ldots, x_n^j)(j = 1, 2, \ldots, N)$ is the jth pattern of the whole dataset, N is the number of the dataset, $\mathrm{d}\sigma_i = (b_1 - a_1)(b_2 - a_2)\ldots(b_n - a_n)/N$, $a_i = \min\{x_i^j, j = 1, 2, \ldots, N\}$, $b_i = \max\{x_i^j, j = 1, 2, \ldots, N\}(1 \leqslant i \leqslant n)$.

3.2. Sensitivity measure of RBFNN with multiple outputs

Considering an RBFNN with multiple outputs, the sensitivity measure can be calculated similarly to the measure for single output network. Assuming that the RBFNN has s outputs, then the output vector of the network can be expressed by $(y_1, y_2, \ldots, y_s)$. Let $w_k = (w_{k1}, w_{k2}, \ldots, w_{kn})^T$ be the kth weight vector connected to the kth output. Thus, each output can be calculated by

$$y_k = \sum_{j=1}^{m} w_{kj} \left(\exp\left(\frac{\sum_{i=1}^{n}(x_i - u_{ji})^2}{-2v_j^2} \right) \right), \quad k = 1, 2, \ldots, s.$$

In this situation, the partial derivatives of the network with respect to the input x_i are denoted by the vector $(\partial y_1/\partial x_i, \partial y_2/\partial x_i, \ldots, \partial y_s/\partial x_i)$. The derivatives can be denoted by Eq. (3.3) via replacing the w_j with w_{kj}. That is

$$\frac{\partial y_k}{\partial x_i} = \sum_{j=1}^{m} w_{kj} \exp\left(\frac{\sum_{k=1}^{n}(x_k - u_{kj})^2}{-2v_j^2} \right) \times \left(\frac{x_i - u_{ij}}{-v_j^2} \right), \quad k = 1, 2, \ldots, s.$$

According to Theorem 3.1, the expectation of the derivative vector

$$e_i = \left(E\left(\frac{\partial y_1}{\partial x_i}\right), E\left(\frac{\partial y_2}{\partial x_i}\right), \ldots, E\left(\frac{\partial y_s}{\partial x_i}\right) \right)$$

can be calculated by Eq. (3.5). We would like to measure the sensitivity of the network with respect to the input by a real number, not by a vector. There are many ways to measure the vector, for example, the following three types:

(1) Length of the vector: $s_i = \|e_i\|$.
(2) Maximum of the vector: $s_i = \max\left\{E\left(\frac{\partial y_1}{\partial x_i}\right), E\left(\frac{\partial y_2}{\partial x_i}\right), \ldots, E\left(\frac{\partial y_s}{\partial x_i}\right)\right\}$.
(3) Minimum of the vector: $s_i = \min\left\{E\left(\frac{\partial y_1}{\partial x_i}\right), E\left(\frac{\partial y_2}{\partial x_i}\right), \ldots, E\left(\frac{\partial y_s}{\partial x_i}\right)\right\}$.

Thus it gives the sensitivity measure of the multiple-output network with respect to each input.

4. Simulation and comparison

4.1. Pretreatment

It is noted that the sensitivity measure (3.1) depends on the joint distribution. In practice, various situations will be encountered. Therefore pretreatment of the given information is needed for numerical simulations. The situations encountered are perhaps classified into three cases. Different pretreatment is carried out for different cases.

- A complete sample dataset is available and its distribution is supposed to be known. In this ideal case, Eq. (3.5) (for normal distribution) or another formula (for another distribution) can be applied directly. For simulation, the distribution mean and variance will be got directly from the distribution. The training dataset composed of N samples will be drawn according to the distribution.
- A complete sample dataset is available, but its distribution is unknown. In this case, based on statistical estimation theory, we may first assume the variable obey some distribution (such as poisson, exponent, normality), then use the database to test the hypotheses, and finally decide whether the hypotheses of the distribution can be receivable or not. The calculating formula will be derived according to the gained distribution information. For simulation, the training dataset will be the given sample dataset. The distribution mean and variance will be given by the sample mean and variance based on the training dataset.
- A sample dataset is not available but only the empirical mean and variance are given. In this case, if no any additional information is available, we are mere to suppose a normal distribution and use formula Eq. (3.5) by substituting empirical mean and variance for distribution mean and variance within Eq. (3.5), and then convert to the first case. That is, for simulation, the distribution mean and variance will be got directly from the distribution and the training dataset composed of N samples will be drawn according to the distribution.

4.2. Rule for selecting the redundant features

We use a simulation to approximate a real function to show the feasibility and effectiveness of our proposed sensitivity measure application to redundant feature reduction.

Let $f(x_1, x_2, \ldots, x_n)$ be a given real differentiable function with n real variables, and $(\xi_1, \xi_2, \ldots, \xi_n)$ be n independently distributed random variables. Their mathematical expectations $(\mu_1, \mu_2, \ldots, \mu_n)$ and variances $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ can be obtained from their distribution information. Consider the random variable $f(\xi_1, \xi_2, \ldots, \xi_n)$. Obviously the equality $\sigma_j^2 = 0$ (for some certain j) implies that the jth variable ξ_j will take the constant value μ_j. The constant in a (random) function has no important impact on the function's structure and properties. Therefore, we can regard the variable (feature) being constant in a function as redundant. Based on this idea, we can think that a random variable (feature) with very small variance is considered as redundant.

Training patterns (training data set) can be drawn from the random variable $f(\xi_1, \xi_2, \ldots, \xi_n)$. Suppose that the N drawn patterns are represented as

$$(x_{ij} | j = 1, 2, \ldots, n; y_i), \quad i = 1, 2, \ldots, N,$$

where $(x_{i1}, x_{i2}, \ldots, x_{in})$ is a sample coming from the distribution of random variable ξ_i and $y_i = f(x_1, x_2, \ldots, x_n)$. If ξ_i has a very small variance, then according to the above discussions we consider it as a redundant feature.

Using the N drawn patterns, we can train a network to approximate the function $f(x_1, x_2, \ldots, x_n)$. Suppose that some features have very small variances and therefore are determined as redundant for the function $f(x_1, x_2, \ldots, x_n)$. The problem is whether we can detect these redundant features by using our sensitivity formula of networks.

When the network training is done, we select the criteria for pruning inputs similar to the criteria mentioned in [6].

First, we calculate sensitivities of all features according to the corresponding formulae. Those sensitivities are then ranked in ascending order,

$$S_{(i)} \leqslant S_{(i+1)}, \quad i = 1, 2, \ldots, n-1.$$

Second, calculate the sensitivity gap between the two neighbor features,

$$g_i = \frac{S_{(i+1)}}{S_{(i)}}, \quad i = 1, 2, \ldots, n-1.$$

Then find the largest gap $g_{i_{\max}}$ and the second largest gap $g_{i_{\max\,\mathrm{II}}}$. If $g_{i_{\max}}$ is much bigger than $g_{i_{\max\,\mathrm{II}}}$, i.e. $Cg_{i_{\max}} > g_{i_{\max\,\mathrm{II}}}$, where C is chosen arbitrarily within the reasonable range, e.g. $C = 0.5$, the features whose sensitivities are smaller than $S_{(i_{\max})}$ are thus selected to the pruning candidate inputs.

Given a real differentiable function $f(x_1, x_2, \ldots, x_n)$ with n real variables and a random vector $(\xi_1, \xi_2, \ldots, \xi_n)$ following some known distribution, we now list our experimental procedure as follows.

Step 1: Get N patterns according to the pretreatments mentioned in Section 4.1.
Step 2: Get the mathematical expectations and the variances for all features.
Step 3: Compute the target output according to the function $f(x_1, x_2, \ldots, x_n)$ for each pattern.
Step 4: Construct one network and train it with the N drawn patterns and their target outputs.
Step 5: Calculate the sensitivities according to the corresponding formulae.
Step 6: Sort the sensitivities in ascending order.
Step 7: Calculate the sensitivity gap of the sorted features.
Step 8: Find the largest gap $g_{i_{\max}}$ and the second largest gap $g_{i_{\max\,\mathrm{II}}}$.
Step 9: Determine the candidate features for deletion according to the pruning criteria mentioned above.
Step 10: Delete the candidate features and check the testing accuracy.

4.3. Simulations

Following the experiment procedure steps 1–10, a number of experiments have been conducted to illustrate the new sensitivity measure application to feature reduction. Here we report the experimental results related to the approximation problem using one RBFNN and the comparison with sensitivity measure proposed in [6]. The sensitivity is calculated according to (3.5).

Consider the following function:

$$f = \sin(\tan(x_1^2) - x_2x_3) - x_4x_5 + \exp(x_6 \sin x_7 + x_8) + x_9 - x_{10}^3 \tag{4.1}$$

which is the function we will construct one RBFNN to approximate. There are 10 features in the function. We can assume some of them to have very small variance. Therefore, according to the above discussion, these features are considered as redundant features existing in the dataset.

For example, x_3, x_6 and x_9 are assumed to have very small variance (0.001). Based on a joint normal distribution where x_3, x_6 and x_9 have very small variances, we can randomly draw 200 patterns for training and 50 patterns for testing and then construct a RBFNN to approximate the given function (4.1). The simulation is then conducted according to the above procedure steps (1)–(10) where the sensitivities for inputs are calculated by the formulae both (3.5) and (2.2), respectively. The intermediate results of the experiment are show in Table 1 where (a) is the experimental results by formula (3.5) and (b) is by formula (2.2).

The parameter C in the experiments is set to be $C = 0.5$. The numbers in bold are the largest gaps and the numbers in italic are the second largest gaps. From (a), we can get $g_{i_{\max\,\mathrm{II}}}/g_{i_{\max}} = 0.022521 < 0.5$. Thus the first three features are selected to be the pruning candidates. After deleting the three features, the network keeps the testing accuracy MSE = 6.3061 the same as before deletion. From (b), we can get $g_{i_{\max\,\mathrm{II}}}/g_{i_{\max}} = 0.8999 > 0.5$. Thus no feature can be

Table 1
Feature sensitivities by two sensitivity measures

Feature	Sensitivity	Gap
(a) Sensitivity by (3.5)		
6	0.008718	1.0008
9	0.008725	1.0008
3	0.008732	143.32
2	1.2514	1.0757
5	1.3461	1.0396
4	1.3995	1.2312
10	1.723	1.0934
7	1.8839	1.1712
1	2.2063	***3.2277***
8	7.1213	0
(b) Sensitivity by (2.2)		
3	0.10606	1.5777
6	0.16733	1.1615
9	0.19434	**2.1335**
2	0.41463	1.0699
4	0.44361	1.0878
5	0.48257	1.3988
10	0.67501	1.0226
7	0.69025	1.256
1	0.86692	***1.9199***
8	1.6644	0

pruned. Correspondingly, we get the comparison results of feature selection shown in bold in the first row of Table 2. In this sense, the measure of sensitivity by using formula (3.5) is more meaningful than by using (2.2).

One may argue that the method given by sensitivity (2.2) also can delete the three redundant features if we choose the parameter $C = 0.02$. It is right. But such a selection of the parameter C is too ad hoc. It seems to be unrealistic. One can mention that the order of features' sensitivity in (a) is almost same as in (b), but the largest gap in (a) is more significant than in (b). This may result from the fact that all simulation data are drawn according to normal distributions.

Similarly, we assume different groups of features being the redundant features existing in the dataset. The final comparing results are shown in Table 2. Due to the randomly drawn data and the trained RBFNN, we get different results for selecting the same group of redundant features. However, it can be still seen that the new proposed measure of sensitivity performs better than the measure proposed by Zurada et al in [6].

4.4. *Experimental results on realistic applications*

In this section, we applied the sensitivity measure to 4 realistic or benchmark datasets to check the effectiveness of the sensitivity formula Eq. (3.5) for deleting redundant attributes in RBFNNs. Intelligent Image-Text Processing dataset is obtained from the Intelligent Image-Text Processing Lab of Hebei University [17]. It is to recognize the font of a given text image. By Gobar filtering, a text image can yield a 24-dimensional real vector. Totally, there are 4 font classes. A RBFNN is trained on the dataset to help recognize the font of a given vector. Another 3 datasets (MPG, Pima, and Breast Cancer) are selected from the frequently-used UCI machine learning repository.

The experiments are conducted according to the following steps. (n is the number of the attributes and s is the number of the classes.)

Step 1: Split the dataset into 2 parts randomly, 70% for training and 30% for testing.

Step 2: Train an RBFNN with n inputs and s outputs by BP algorithm.

Step 3: Test the trained RBFNN.

Step 4: Compute the sensitivity for each of the n attributes.

Step 5: Delete the redundant attributes determined by the rule given in Section 4.2.

Step 6: Re-train an RBFNN with the remaining inputs and s outputs by BP algorithm.

Step 7: Re-test the new RBFNN.

Step 8: Repeat the above steps 10 times and take the average of training and testing accuracy.

The deleted redundant attributes change with different splits of the datasets. For example, we consider the first dataset, Intelligent Image-Text Processing dataset. During the 10 experiments, attributes 2, 6, and 21 are deleted 8 times, and attributes 3 and 20 are deleted 6 times. It is worth noting that, for each experiment, there is a little difference among the selected redundant features, and therefore, we take the averaged training/testing accuracy in Step 8 above.

The averaged training and testing accuracy before and after deleting the selected redundant attributes is summarized in Table 3. The experimental results indicate that the testing accuracy has not decreased after deleting attributes with small sensitivities. They illustrate the effectiveness of the new sensitivity measure to feature selection.

5. Conclusion

Sensitivity analysis of the input is an efficient way to simplify the structure of neural networks, especially the structure of RBFNNs. The sensitivity-based feature selection (redundant feature deletion) can improve the structure, performance, calculating time, etc. of the network.

Zurada et al. in [6] proposed a practical and popular measure of sensitivity that is defined as a matrix of the partial derivatives of the network output to its inputs. The sensitivity of each input is evaluated based on the entire training and testing data set. The largest gap is used in the pruning criteria.

Table 2
Features selected using the sensitivity analysis based on different measures

Redundant features existing	By measure (2.2)		By new measure (3.5)	
	Number of deleted features	Deleted features	Number of deleted features	Deleted features
3,6,9	**3**	**6,3,9**	**3**	6,9,**3**
3,6,9	0	None	3	6,9,3
2,9	2	2,9	2	2,9
3,5,7,9	0	None	1	7
1,3,5,7	4	1,7,5,3	4	1,7,5,3
2,3,5	0	None	3	5,3,2
2,3,5	3	3,2,5	3	3,2,5

Table 3
Averaged training and testing accuracy

Datasets	Training accuracy		Testing accuracy	
	Before deleting	After deleting	Before deleting	After deleting
Image-text processing	0.96	0.95	0.95	0.97
MPG	0.8167	0.7916	0.7071	0.7518
Pima	0.7412	0.7420	0.7128	0.7516
Breast cancer	0.9764	0.9875	0.9211	0.9385

In this paper, we started with proposing the definition of derivative of one function with a random variable based on the concept of convergence in probability. The derivative in the probability measure space has been proved to have the same form as the real space. That is, the derivative in the probability space is obtained by replacing the real variables with the random variables inside the real derivative formula. The definition of derivative is extended easily to the partial derivatives in the probability measure space. Taking inputs of the network as random variables, this paper extends the sensitivity measure in [6] from the real space to the probability measure space. The sensitivity formulae are deduced for RBFNNs as (3.5) which depends only on the numerical characteristics of the input random variables with normal distribution. Simulations are conducted to illustrate the effectiveness and universality of the sensitivity analysis for RBFNNs in the probability measure space. The pruning criteria are similar to that in [6] for the purpose of comparison.

It is noted that the sensitivity formula (3.5) is deduced based on the normally distributed variables. If the distribution changes, we may deduce the different forms of sensitivity formulae.

Acknowledgments

This paper is supported by National Natural Science Foundation of China (Project No. 60473045) and Hebei Province High Technology Project (Project No. 04213533).

The second author would like to deliver her sincere appreciation to her colleague Zhang Hui for his helpful suggestions. Thank Li Qian for her further experiment conduction.

Appendix A. Proof of Theorem 4.1

From (4.1), we can obtain

$$
\begin{aligned}
(f'_{(i)})^2 &= \left(\sum_{j=1}^{m} w_j \exp\left(\frac{\Sigma_{k=1}^{n}(\xi_k-u_{kj})^2}{-2v_j^2}\right)\left(\frac{\xi_i-u_{ij}}{-v_j^2}\right)\right)^2 \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \exp\left(-\frac{1}{2}\frac{\Sigma_{k=1}^{n}(\xi_k-u_{kj})^2}{v_j^2}-\frac{1}{2}\frac{\Sigma_{k=1}^{n}(\xi_k-u_{kt})^2}{v_t^2}\right)\times\left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right) \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right)\times\exp\left(\sum_{k=1}^{n}-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right) \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right)\times\prod_{k=1}^{n}\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right).
\end{aligned}
$$

Notice that the random variables $\xi_1,\xi_2,\ldots,\xi_n$ are independent, we have

$$
\begin{aligned}
E\left[(f'_{(i)})^2\right] = E\Bigg[&\sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right) \\
&\times\prod_{k=1}^{n}\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right)\Bigg]
\end{aligned}
$$

$$= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t E\left[\left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2 v_t^2}\right)\right.$$
$$\left.\times \exp\left(-\frac{1}{2}\left(\frac{(\xi_i-u_{ij})^2}{v_j^2}+\frac{(\xi_i-u_{it})^2}{v_t^2}\right)\right)\right]$$
$$\times \prod_{k=1,k\neq i}^{n} E\left[\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right)\right].$$

Recalling that the random vector (i.e., all of the random variables) is normally distributed with means $(\mu_1,\mu_2,\ldots,\mu_n)$ and variances $(\sigma_1^2,\sigma_2^2,\ldots,\sigma_n^2)$, the density function is

$$f_{\xi_k}(x_k)=\frac{1}{\sqrt{2\pi}\sigma_k}\exp\left(-\frac{1}{2}\frac{(x_k-\mu_k)^2}{\sigma_k^2}\right).$$

Considering $\int_{-\infty}^{\infty} f(x_k)\,\mathrm{d}x_k=1$ holds true for any density function, we get

$$E\left[\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right)\right]$$
$$=\frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{(x_k-u_{kj})^2}{v_j^2}+\frac{(x_k-u_{kt})^2}{v_t^2}\right)\right)\exp\left(-\frac{1}{2}\frac{(x_k-\mu_k)^2}{\sigma_k^2}\right)\mathrm{d}x_k$$
$$=\frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{(x_k-u_{kj})^2}{v_j^2}+\frac{(x_k-u_{kt})^2}{v_t^2}+\frac{(x_k-\mu_k)^2}{\sigma_k^2}\right)\right)\mathrm{d}x_k$$
$$=\frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{v_t^2\sigma_k^2(x_k-u_{kj})^2+v_j^2\sigma_k^2(x_k-u_{kt})^2+v_j^2v_t^2(x_k-\mu_k)^2}{v_j^2v_t^2\sigma_k^2}\right)\right)\mathrm{d}x_k. \tag{A.1}$$

Consider the term above the line in the fraction inside integral (A.1), we have

$$v_t^2\sigma_k^2(x_k-u_{kj})^2+v_j^2\sigma_k^2(x_k-u_{kt})^2+v_j^2v_t^2(x_k-\mu_k)^2$$
$$=v_t^2\sigma_k^2(x_k^2-2x_ku_{kj}+u_{kj}^2)+v_j^2\sigma_k^2(x_k^2-2x_ku_{kt}+u_{kt}^2)+v_j^2v_t^2(x_k^2-2x_k\mu_k+\mu_k^2)$$
$$=x_k^2(v_t^2\sigma_k^2+v_j^2\sigma_k^2+v_j^2v_t^2)-2x_k(v_t^2\sigma_k^2u_{kj}+v_j^2\sigma_k^2u_{kt}+v_j^2v_t^2\mu_k)+(v_t^2\sigma_k^2u_{kj}^2+v_j^2\sigma_k^2u_{kt}^2+v_j^2v_t^2\mu_k^2)$$
$$=a_kx_k^2-2b_kx_k+c_k$$
$$=a_k\left(x_k-\frac{b_k}{a_k}\right)^2+c_k-\frac{b_k^2}{a_k},$$

where

$$a_k=v_t^2\sigma_k^2+v_j^2\sigma_k^2+v_j^2v_t^2,$$

$$b_k=v_t^2\sigma_k^2u_{kj}+v_j^2\sigma_k^2u_{kt}+v_j^2v_t^2\mu_k,$$

$$c_k=v_t^2\sigma_k^2u_{kj}^2+v_j^2\sigma_k^2u_{kt}^2+v_j^2v_t^2\mu_k^2.$$

Replace the corresponding terms in (A.1) with a_k,b_k,c_k, we get

$$E\left[\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right)\right]$$
$$=\frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{a_k(x_k-(b_k/a_k))^2+c_k-(b_k^2/a_k)}{v_j^2v_t^2\sigma_k^2}\right)\right)\mathrm{d}x_k$$

$$
\begin{aligned}
&= \frac{1}{\sqrt{2\pi}\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{(x_k-(b_k/a_k))^2}{v_j^2v_t^2\sigma_k^2/a_k}\right)-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2v_t^2\sigma_k^2}\right)\right)\mathrm{d}x_k \\
&= \frac{v_jv_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2v_t^2\sigma_k^2}\right)\right)\frac{1}{\sqrt{2\pi a_k}v_jv_t\sigma_k}\int_{-\infty}^{\infty}\exp\left(-\frac{1}{2}\left(\frac{(x_k-(b_k/a_k))^2}{v_j^2v_t^2\sigma_k^2/a_k}\right)\right)\mathrm{d}x_k \\
&= \frac{v_jv_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2v_t^2\sigma_k^2}\right)\right).
\end{aligned}
\tag{A.2}
$$

Now, we evaluate the following mathematical expectation:

$$
\begin{aligned}
&E\left[\left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right)\exp\left(-\frac{1}{2}\left(\frac{(\xi_i-u_{ij})^2}{v_j^2}+\frac{(\xi_i-u_{it})^2}{v_t^2}\right)\right)\right] \\
&= \frac{1}{\sqrt{2\pi}\sigma_i}\int_{-\infty}^{\infty}\left(\frac{(x_i-u_{ij})(x_i-u_{it})}{v_j^2v_t^2}\right)\exp\left(-\frac{1}{2}\left(\frac{(x_i-u_{ij})^2}{v_j^2}+\frac{(x_i-u_{it})^2}{v_t^2}\right)\right) \\
&\quad\times\exp\left(-\frac{1}{2}\frac{(x_i-\mu_i)^2}{\sigma_i^2}\right)\mathrm{d}x_i \\
&= \frac{v_jv_t}{\sqrt{a_i}}\exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2v_t^2\sigma_i^2}\right)\right)\frac{1}{\sqrt{2\pi a_i}v_jv_t\sigma_i}\int_{-\infty}^{\infty}\left(\frac{(x_i-u_{ij})(x_i-u_{it})}{v_j^2v_t^2}\right)\exp\left(-\frac{1}{2}\left(\frac{(x_i-(b_i/a_i))^2}{v_j^2v_t^2\sigma_i^2/a_i}\right)\right)\mathrm{d}x_i \\
&= \frac{1}{v_jv_t\sqrt{a_i}}\exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2v_t^2\sigma_i^2}\right)\right)\frac{1}{\sqrt{2\pi a_i}v_jv_t\sigma_i}\int_{-\infty}^{\infty}(x_i^2-(u_{ij}+u_{it})x_i+u_{ij}u_{it})\exp\left(-\frac{1}{2}\left(\frac{(x_i-(b_i/a_i))^2}{v_j^2v_t^2\sigma_i^2/a_i}\right)\right)\mathrm{d}x_i \\
&= \frac{1}{v_jv_t\sqrt{a_i}}\exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2v_t^2\sigma_i^2}\right)\right)(A+B+C),
\end{aligned}
$$

where

$$
\begin{aligned}
A &= \frac{1}{\sqrt{2\pi a_i}v_jv_t\sigma_i}\int_{-\infty}^{\infty}x_i^2\exp\left(-\frac{1}{2}\left(\frac{(x_i-(b_i/a_i))^2}{v_j^2v_t^2\sigma_i^2/a_i}\right)\right)\mathrm{d}x_i \\
&= \frac{v_j^2v_t^2\sigma_i^2}{a_i}+\left(\frac{b_i}{a_i}\right)^2, \\
B &= -(u_{ij}+u_{it})\frac{1}{\sqrt{2\pi a_i}v_jv_t\sigma_i}\int_{-\infty}^{\infty}x_i\exp\left(-\frac{1}{2}\left(\frac{(x_i-(b_i/a_i))^2}{v_j^2v_t^2\sigma_i^2/a_i}\right)\right)\mathrm{d}x_i = -(u_{ij}+u_{it})\frac{b_i}{a_i}, \\
C &= \frac{1}{\sqrt{2\pi a_i}v_jv_t\sigma_i}\int_{-\infty}^{\infty}u_{ij}u_{it}\exp\left(-\frac{1}{2}\left(\frac{(x_i-(b_i/a_i))^2}{v_j^2v_t^2\sigma_i^2/a_i}\right)\right)\mathrm{d}x_i = u_{ij}u_{it}.
\end{aligned}
$$

It is followed by

$$
\begin{aligned}
&E\left[\left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2v_t^2}\right)\exp\left(-\frac{1}{2}\left(\frac{(\xi_i-u_{ij})^2}{v_j^2}+\frac{(\xi_i-u_{it})^2}{v_t^2}\right)\right)\right] \\
&= \frac{1}{v_jv_t\sqrt{a_i}}\exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2v_t^2\sigma_i^2}\right)\right)\left(\frac{v_j^2v_t^2\sigma_i^2}{a_i}+\left(\frac{b_i}{a_i}\right)^2-(u_{ij}+u_{it})\frac{b_i}{a_i}+u_{ij}u_{it}\right) \\
&= \frac{1}{v_jv_t\sqrt{a_i}}\exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2v_t^2\sigma_i^2}\right)\right)\left(\frac{v_j^2v_t^2\sigma_i^2}{a_i}+\left(\frac{b_i}{a_i}-u_{ij}\right)\left(\frac{b_i}{a_i}-u_{it}\right)\right).
\end{aligned}
\tag{A.3}
$$

Replacing the corresponding items in $E\left[(f'_{(i)})^2\right]$ with (A.2) and (A.3), we then get

$$\begin{aligned}
E\left[(f'_{(i)})^2\right] &= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t E\left[\left(\frac{(\xi_i-u_{ij})(\xi_i-u_{it})}{v_j^2 v_t^2}\right)\right. \\
&\quad \left.\times \exp\left(-\frac{1}{2}\left(\frac{(\xi_i-u_{ij})^2}{v_j^2}+\frac{(\xi_i-u_{it})^2}{v_t^2}\right)\right)\right]\prod_{k=1,k\neq i}^{n} E\left[\exp\left(-\frac{1}{2}\left(\frac{(\xi_k-u_{kj})^2}{v_j^2}+\frac{(\xi_k-u_{kt})^2}{v_t^2}\right)\right)\right] \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \frac{1}{v_j v_t \sqrt{a_i}} \exp\left(-\frac{1}{2}\left(\frac{c_i-(b_i^2/a_i)}{v_j^2 v_t^2 \sigma_i^2}\right)\right)\left(\frac{v_j^2 v_t^2 \sigma_i^2}{a_i}+\left(\frac{b_i}{a_i}-u_{ij}\right)\left(\frac{b_i}{a_i}-u_{it}\right)\right) \\
&\quad \times \prod_{k=1,k\neq i}^{n}\left[\frac{v_j v_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2 v_t^2 \sigma_k^2}\right)\right)\right] \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \frac{1}{v_j^2 v_t^2}\left(\frac{v_j^2 v_t^2 \sigma_i^2}{a_i}+\left(\frac{b_i}{a_i}-u_{ij}\right)\left(\frac{b_i}{a_i}-u_{it}\right)\right)\prod_{k=1}^{n}\left[\frac{v_j v_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2 v_t^2 \sigma_k^2}\right)\right)\right] \\
&= \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t \left(\frac{\sigma_i^2}{a_i}+\frac{1}{v_j^2 v_t^2}\left(\frac{b_i}{a_i}-u_{ij}\right)\left(\frac{b_i}{a_i}-u_{it}\right)\right)\prod_{k=1}^{n}\left[\frac{v_j v_t}{\sqrt{a_k}}\exp\left(-\frac{1}{2}\left(\frac{c_k-(b_k^2/a_k)}{v_j^2 v_t^2 \sigma_k^2}\right)\right)\right]
\end{aligned}$$

which shows that formula (3.5) given in Theorem 4.1 is correct, and therefore, completes the proof.

Appendix B. The calculating formulae of uniformly distributed random variables

Given independently and uniformly distributed random variables $\xi_1, \xi_2, \ldots, \xi_n$, the density function is respectively given by

$$\varphi(x_k) = \begin{cases} \frac{1}{f_k - e_k}, & e_k \le x_k \le f_k \\ 0, & \text{else} \end{cases} \qquad 1 \leqslant k \leqslant n.$$

The sensitivity defined by (3.1) is as the following:

$$E\left[(f'_{(i)})^2\right] = \sum_{j=1}^{m}\sum_{t=1}^{m} w_j w_t A_i \prod_{k=1,k\neq i}^{n} A_k, \tag{B.1}$$

where

$$\begin{aligned}
A_k &= \frac{1}{f_k - e_k}\exp\left(-\frac{1}{2}\left((u^2v)_{jt} - v_{jt}((uv)_{jt})^2\right)\right) \\
&\quad \times \left\{ \begin{aligned} &\frac{h}{2}\left[\begin{aligned} &\exp\left(-\tfrac{1}{2}\left(v_{jt}(e_k-(uv)_{jt})^2\right)\right)+\exp\left(-\tfrac{1}{2}\left(v_{jt}(f_k-(uv)_{jt})^2\right)\right) \\ &+2\sum_{l=1}^{N-1}\exp\left(-\tfrac{1}{2}\left(v_{jt}(e_k+lh-(uv)_{jt})^2\right)\right) \end{aligned}\right] \\ &-\tfrac{Nh^3}{12}\left(2v_{jt}(d_k-(uv)_{jt})^2-2\right)\exp\left(-\tfrac{1}{2}\left(v_{jt}(d_k-(uv)_{jt})^2\right)\right) \end{aligned}\right\} \\
&\quad (1 \leqslant k \leqslant n, \quad k \neq i), \\
A_i &= \frac{1}{v_j^2 v_t^2 (f_i - e_i)}\exp\left(-\frac{1}{2}\left((u^2v)_{jt} - v_{jt}((uv)_{jt})^2\right)\right) \\
&\quad \times \left\{ \begin{aligned} &\frac{h}{2}\left[\begin{aligned} &(e_i-u_{ij})(e_i-u_{it})\exp\left(-\tfrac{1}{2}\left(v_{jt}(e_k-(uv)_{jt})^2\right)\right)+(f_i-u_{ij})(f_i-u_{it})\exp\left(-\tfrac{1}{2}\left(v_{jt}(f_k-(uv)_{jt})^2\right)\right) \\ &+2\sum_{l=1}^{N-1}(e_i+lh-u_{ij})(e_i+lh-u_{it})\exp\left(-\tfrac{1}{2}\left(v_{jt}(e_k+lh-(uv)_{jt})^2\right)\right) \end{aligned}\right] \\ &-\tfrac{lh^3}{12}\exp\left(-\tfrac{1}{2}\left(v_{jt}(x_i-(uv)_{jt})^2\right)\right)\left[\begin{gathered} -v_{jt}x_i^3 + x_i^2(-v_{jt})(2-(u_{ij}+u_{it})-v_{jt}) \\ +x_i(2-v_{jt}u_{ij}u_{it}-(uv)_{jt}+(v_{jt}-(uv)_{jt})(u_{ij}+u_{it})-(v_{jt})^2(uv)_{jt}) \\ -v_{jt}(uv)_{jt}(u_{ij}u_{it}-(u_{ij}+u_{it})+(uv)_{jt})+(2-(u_{ij}+u_{it})-v_{jt}) \end{gathered}\right] \end{aligned}\right\},
\end{aligned}$$

$v_{jt} = \frac{v_t^2+v_j^2}{v_j^2 v_t^2}$, $(uv)_{jt} = \frac{v_t^2 u_{kj}+v_j^2 u_{kt}}{v_t^2+v_j^2}$, $(u^2v)_{jt} = \frac{v_t^2 u_{kj}^2+v_j^2 u_{kt}^2}{v_t^2+v_j^2}$. The above formula holds for some $d_k \in [e_k, f_k]$ $(1 \leqslant k \leqslant n)$ [15].

References

[1] A. Alexandridis, H. Sarimveis, G. Bafas, A new algorithm for online structure and parameter adaptation of RBF networks, Neural networks 16 (2003) 1003–1017.

[2] N.B. Karayiannis, Reformulated radial basis neural networks trained by gradient decent, IEEE Transactions on Neural Networks 10 (3) (1999) 657–671.

[3] X. Wang, C. Li, A new definition of sensitivity for RBFNN and its applications to feature reduction, in Second International Symposium on Neural Networks, LNCS 3496, pp. 81–86, 2005.

[4] W.Y.N.G. Wing, D.S. Yeung, Input dimensionality reduction for Radial Basis Neural Network classification problems using sensitivity analysis, in: Proceedings of the First International Conference on Machine Learning and Cybernetics, 2002, pp. 2214–2219.

[5] D. Shi, D.S. Yeung, J. Gao, Sensitivity analysis applied to the construction of radial basis function networks, Neural Networks 18 (2005) 951–957.

[6] J.M. Zurada, A. Malinowski, S. Usui, Perturbation method for deleting redundant inputs of perceptron networks, Neurocomputing 14 (1997) 177–193.

[7] R.M. Dudley, Real Analysis and Probability, Cambridge University Press, Cambridge, New York, 2002.

[8] P.H. Garthwaite, Statistical Inference, Oxford University Press, Oxford, New York, 2002.

[9] P.R. Halmos, Measure Theory, New York, 1950.

[10] J.Y. Choi, C.-H. Choi, Sensitivity analysis of multilayer perceptron with differentiable activation functions, IEEE Transactions on Neural Networks 3 (1) (1992) 101–107.

[11] X. Zeng, D.S. Yeung, Sensitivity analysis of multilayer perceptron to input and weight perturbations, IEEE Transactions on Neural Networks 12 (6) (2001) 1358–1366.

[12] S.W. Piché, The selection of weight accuracies for madalines, IEEE Transactions on Neural Networks 6 (2) (1995) 432–445.

[13] L. Xu, et al., Mathematic Dictionary, ShanXi Educational Publishing Company, Southeast University Publishing Company, Science and Technology Publishing Company of China, Vol. 4, August 2002, p. 371.

[14] Z. Sheng, S. Xie, C. Pan, Probability Theory and Mathematical Statistics, third ed., Higher Education Publishing Company, 2001.12. pp. 241–253.

[15] C. Lin, Numerical Calculating Methodology, Vol. 1, Science Publishing Company. 2000.8. pp. 178–180.

[16] E. Parzen, Stochastic Processes, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[17] http://www.icmlc.org/bsdata.htm.

Xi-Zhao Wang received his B.Sc. and M.Sc. degrees in mathematics from Hebei University, Baoding, China, in 1983 and 1992, respectively, and his Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China, in 1998. From 1983 to 1998, he worked as a Lecturer, an Associate Professor, and a Full Professor in the Department of Mathematics, Hebei University. From 1998 to 2001, he worked as a Research Fellow at the Department of Computing, Hong Kong Polytechnic University, Kowloon. Since 2001, he has been the Dean and Professor of the Faculty of Mathematics and Computer Science, Hebei University. His main research interests include inductive learning with fuzzy representation, fuzzy measures and integrals, neuro-fuzzy systems and genetic algorithms, feature extraction, multi-classifier fusion, and applications of machine learning. So far, he has published over 40 international journal papers. He is an IEEE senior member and is an associate editor of IEEE Transactions on SMC part B, and is an associate editor of International Journal of Information Science.

Chun-Guo Li received her B.Sc. and M.Sc. degrees in mathematics application from Hebei University, Baoding, China, in 2003 and 2006, respectively. Her research interests focus on the sensitivity analysis of Radial Basis Function Neural Networks

Daniel S. Yeung (M'89–SM'99) received his Ph.D. degree in applied mathematics from Case Western Reserve University, Cleveland, OH, in 1974. In the past, he has worked as an Assistant Professor of Mathematics and Computer Science at Rochester Institute of Technology, Rochester, NY, as a Research Scientist in the General Electric Corporate Research Center, and a System Integration Engineer at TRW, Inc. Currently, he is a Chair Professor of Computing, The Hong Kong Polytechnic University, Hong Kong. His current research interests include neural-network sensitivity analysis, expert neural-network hybrid systems, off-line handwritten Chinese character recognition, and fuzzy expert systems. Dr. Yeung was the President of the IEEE Computer Chapter of Hong Kong for 1991 and 1992. He now chairs the technical committees on Cybernetics of the IEEE Systems, Man, and Cybernetics Society. He is also an Associate Editor for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS and the IEEE TRANSACTIONS ON NEURAL NETWORKS.