Applied Soft Computing 15 (2014) 57–66

Contents lists available at ScienceDirect

Applied Soft Computing

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

A new constructive neural network method for noise processing and its application on stock market prediction[☆]

Lu Xi[a], Hou Muzhou[b,*], Moon Ho Lee[c], Jun Li[c], Duan Wei[c], Han Hai[c], Yalin Wu[d]

[a] *Business School, Central South University, Changsha 410083, China*
[b] *School of Mathematics and Statistics, Central South University, Changsha 410083, China*
[c] *Division of Electronic and Information Engineering, College of Engineering, Chonbuk National University, Republic of Korea*
[d] *Department of Smart information systems, Jeonju University, Jeonju, Jeonbuk, Republic of Korea*

ARTICLE INFO

Article history:
Received 13 October 2012
Received in revised form 29 August 2013
Accepted 22 October 2013
Available online 1 November 2013

Keywords:
Neural network architecture
Decay RBF neural networks
Overfitting
Noise
Stock market prediction

ABSTRACT

In this paper, in order to optimize neural network architecture and generalization, after analyzing the reasons of overfitting and poor generalization of the neural networks, we presented a class of constructive decay RBF neural networks to repair the singular value of a continuous function with finite number of jumping discontinuity points. We proved that a function with m jumping discontinuity points can be approximated by a simplest neural network and a decay RBF neural network in $L^2(\mathbb{R})$ by each ε error, and a function with m jumping discontinuity point $y=f(x), \quad x\in E\subset\mathbb{R}^d$ can be constructively approximated by a decay RBF neural network in $L^2(\mathbb{R}^d)$ by each $\varepsilon>0$ error. Then the whole networks will have less hidden neurons and well generalization in the same of the first part. A real world problem about stock closing price with jumping discontinuity have been presented and verified the correctness of the theory.

1. Introduction

Neural networks have attracted increasing attention from researchers in many fields, including information processing, computer science, economics, medicine and mathematics, and have been used to solve a wide range of problems such as data mining, function approximation, pattern recognition, expert system and data prediction, etc. The widespread popularity of neural networks in many fields is mainly due to their ability to approximate complex multivariate nonlinear functions directly from the input samples. Neural networks can provide models for a large class of natural and artificial phenomena that are difficult to handle using classical parametric techniques [1–5].

One of the most important problems that neural network designers face today is choosing an appropriate network size for a given application. However, the process of selecting adequate neural network architecture for a given problem is still a controversial issue.

And it is empirically known that the problem is particularly serious when the size of the network is large. When a network is trained with noisy data, it may have a very small training error that is caused by fitting the noise rather than the true function underlying the data. In such situations, the generalization error tends to be larger than its optimal level because the trained network may deviate from the true function [6]. We also call this phenomenon overfitting [7–9].

The overfitting problem is a critical issue that usually leads to poor generalization [10–12]. One of the main reasons of over-fitting is the excessive noise data or singular value in the practical problems [13,14]. So the traditional methods of processing noise data are to remove noise data before approximation through various algorithms such as wavelet transform [15–18], principal component analysis [19,20] and various filtering algorithms [21–24]. But sometimes, the “noisy” data we think of ways to remove are often some singular values of a process which contains important information [25].

In this paper, in order to optimize neural network architecture and generalization, after analyzing the reasons of poor generalization and overfitting of neural networks, we presented a class of constructive decay RBF neural networks to repair the singular values of a continuous function with finite number of jumping discontinuity points. And a real world problem about stock closing price with a jumping discontinuity have been presented and verified the correctness of the theory. First, we will consider some noise

☆ This work was supported by the Natural Science Foundation of China under Grants 61375063, 60970097 and 11271376.
* Corresponding author. Tel.: +86 13787088322.
E-mail addresses: luke84321@sina.com (L. Xi), houmuzhou@sina.com (H. Muzhou), moonho@chonbuk.ac.kr (M.H. Lee), lijun52018@jbnu.ac.kr (J. Li), sinder0403@gmail.com (D. Wei), hhhtgy@jbnu.ac.kr (H. Hai), wuyalin@hanmail.net (Y. Wu).

1568-4946/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2013.10.013

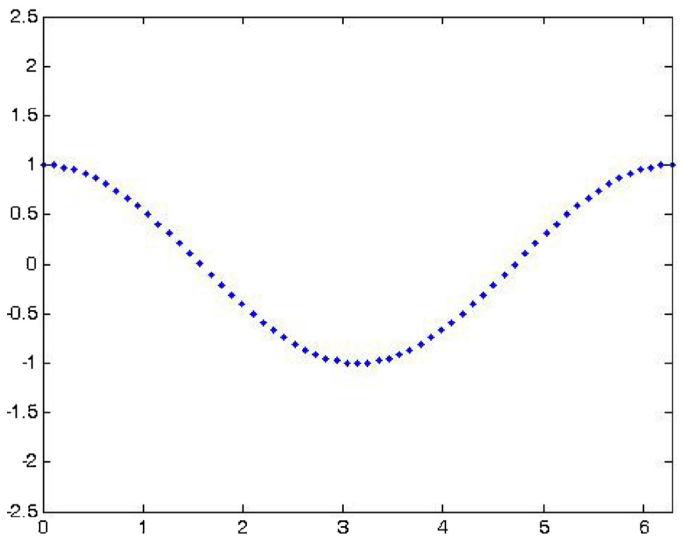

Fig. 1. Sample dataset A comes from function $y=\cos x$.

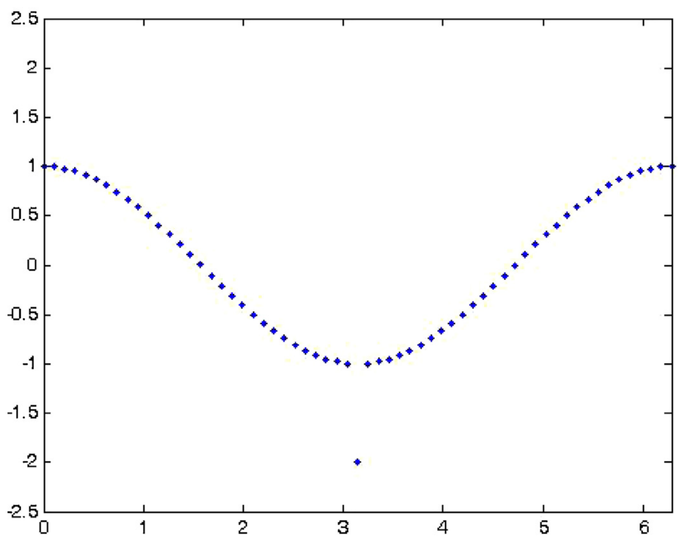

Fig. 2. Sample dataset B comes from function $y=f_1(x)$.

data as singular values of a continuous function - jump discontinuity point. The continuous part can be approximated using less size neural network, which have optimal architecture and good generalization performance, by traditional algorithms such as constructive algorithm for feedforward neural networks with incremental training [26,27], BP algorithm [28,29], ELM algorithm [30,31], various constructive algorithm[32–34], RBF approximation [35–37] and SVM [38]. Then, we will construct a RBF neural network to fit the singular value with every ε error in $L^2(\mathbb{R}^d)$, and then the whole network will have optimal architecture and generalization.

This paper is organized as follows; Section 2 investigates the phenomenon of neural network overfitting caused by noisy data. Section 3 gives some previous works on approximation of functions by neural networks. Section 4 investigates constructive multidimensional approximation of a function with one jump discontinuity point. Theorems 6–9 are proved. Section 5 investigates constructive multidimensional approximation of a function with finite number of jump discontinuity points, Theorems 10–13 are proved. In section 6, a real world problem about stock closing price with a jumping discontinuity point have been presented and verified the correctness of the theory. Section 7 provides some conclusions.

2. The phenomenon of neural networks over-fitting caused by noisy data

Overfitting is a well-known generalization problem for neural network due to the finite training set, which greatly reduce its generalization ability in practical applications [39]. But one of the important factors to cause overfitting is noisy data[40]. In the real world, some underlying function relationship between the input and desired output are simple, but the sample data are always corrupted by noise to some degree. Consequently learning with noisy data would need too many hidden neurons and then results in poor generalization ability. We will describe this phenomenon through a simple experiment.

Example: the following sample dataset A comes from the function $y=\cos x$ which contained 60 points (Fig. 1). And sample dataset B comes from function

$$y=f_1(x)=\begin{cases}\cos x & x\neq\pi\\ -2 & x=\pi\end{cases},$$

which has a noise data in function $y=\cos x$ and also contained 60 points (Fig. 2).

In the following experiments, we will approximate the dataset A and B using traditional single hidden layer feedforward neural networks trained by BP algorithm. Although there are many variants of BP algorithm, a faster BP algorithm called Levenberg–Marquardt is used in our experiments. All the experiments are carried out in MATLAB 7.10 (R2010a) environment running in a Intel(R) Core(TM) i3-2120 3.30 GHz CPU. We will use 54 points of the dataset to train the neural networks and the other 6 points to test the networks.

In the experiments, we will compare the following performance index: training time; the ratio of the training time with the training time of dataset A (RTT)

$$\text{RTT}=\frac{\text{training time of dataset } B}{\text{training time of dataset } A};$$

training and testing root mean square error (RMSE)

$$\text{RMSE}=\sqrt{\frac{1}{n+1}\sum_{i=0}^{n}(f(x_i)-\tilde{y}_i)^2}$$

where $\tilde{y}_i$ is the output value of the neural networks in the simulation point; the ratio of training RMSE (RTRR)

$$\text{RTRR}=\frac{\text{trainging time of dataset } B}{\text{training time of dataset } A}$$

the ratio of testing RMSE (RTER)

$$\text{RTER}=\frac{\text{testing time of dataset } B}{\text{testing time of dataset } A}$$

and the number of the hidden neurons.

In Fig. 3, the neural network only need 4 hidden neurons to fit the function $y=\cos x$ very well with 0.234 CPU training time, 9.2318×10^{-7} training RMSE and 0.00026123 testing RMSE.

In Fig. 4, because there is one noise point in dataset B, 4 hidden neurons in the neural networks is not enough to fit the dataset B, which it spent 5.8926 CPU time to train the networks with 0.012537 training RMSE that is 13580 times of that in Fig. 3, and 0.0057322 testing RMSE which is 21.943 times of that in Fig. 3.

So we increased hidden neurons to 10 to fit it well, in Fig. 5, it spent 2.4024 CPU time to train the networks with 9.9419×10^{-7} training RSME which is almost equals that in Fig. 1, but the testing RMSE is 0.00083815 that is larger 3.2085 times than that in Fig. 1.

Table 1
Performance comparison for dataset *A* with *B*.

Performance/dataset	Training				Testing		Neurons
	Time	RTT	RMSE	RTRR	RMSE	RTER	
A	0.234	1	9.2318×10^{-7}	1	0.00026123	1	4
B	5.8968	25.2	0.012537	13,580	0.0057322	21.943	4
B	2.4024	10.267	9.9419×10^{-7}	1.0769	0.0008318	3.2085	10

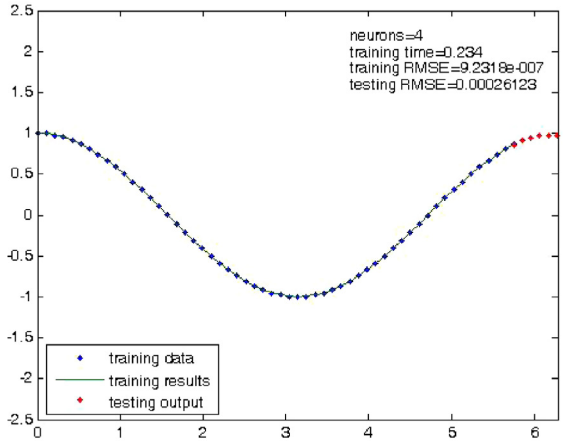

Fig. 3. Training and testing to dataset *A* with 4 neurons.

So with the increase of the complexity of the networks and the precision of training RMSE, the generalization of the neural networks was reduced instead because of the noise data.

The results above three experiments are concluded into the Table 1.

In this paper, the dataset with noisy data is divided into two parts, one part contains simple function relationship, and the other part consists of jumping discontinuity points. The first part can be approximated with the optimal neural network architecture, which has less number of hidden neurons and good generalization performance, by traditional algorithm such as BP, ELM and SVM. At the same time, we will construct a RBF neural network to approximate the singular value with every ε error in $L^2(\mathbb{R}^d)$ which has no influence to the generalization of the first part and the whole neural networks.

3. Previous works on approximation of functions by neural networks

There are many good results on approximation of continuous functions relationship without noisy data by neural networks that have best performing architecture and well generalization by properly using traditional algorithm such BP, ELM, SVM and some constructive approaches.

Let $\phi : \mathbb{R} \to \mathbb{R}$. Define

$$\sum\nolimits^{d}(\phi) \equiv span\left\{\phi(w.x+b) : b \in \mathbb{R}, \quad w, x \in \mathbb{R}^d\right\}, \tag{1}$$

Then $N \in \sum^{d}(\phi)$ if and only if

$$N(x) = \sum_{j=0}^{n} c_j \phi(w_j.x + b_j), \tag{2}$$

where $c_j, \quad j = 0, \cdots, n$ are real numbers and n is a positive integer.

We say that ϕ is a sigmoid function, if it verifies $\lim\limits_{t\to-\infty} \phi(t) = 0$ and $\lim\limits_{t\to\infty} \phi(t) = 1$, then (2) is called single-layer feed-forward neural network. If $\phi(\mathrm{x}) = \varphi(\|\mathrm{x}\text{-}\mathrm{x}_0\|)\ x \in \mathbb{R}^d$, we call $\phi(\mathrm{x})$ RBF function, then (2) is called RBF neural networks.

In the case of continuous functions we have the following density results

Theorem 1. ([41]) Let $\phi \in C(\mathbb{R})$. Then $\sum^{d}(\phi)$ is dense in $C(\mathbb{R}^d)$ in the topology of uniform convergence on compact if and only if ϕ is not a polynomial.

Corollary 1. ([42]) $y = f(x) \in C(\mathbb{R}^d)$ can be approximated by a simplest neural networks (such as with minimum number of hidden neurons).

In the case of not necessarily continuous functions we also have some density results.

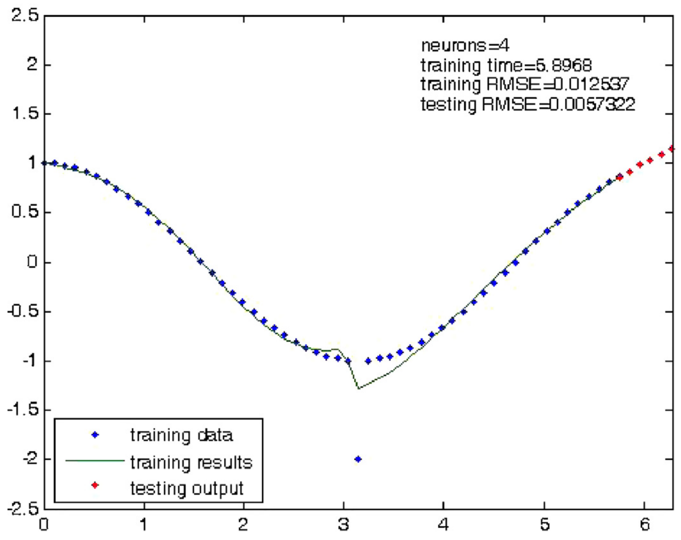

Fig. 4. Training and testing to dataset *B* with 4 neurons.

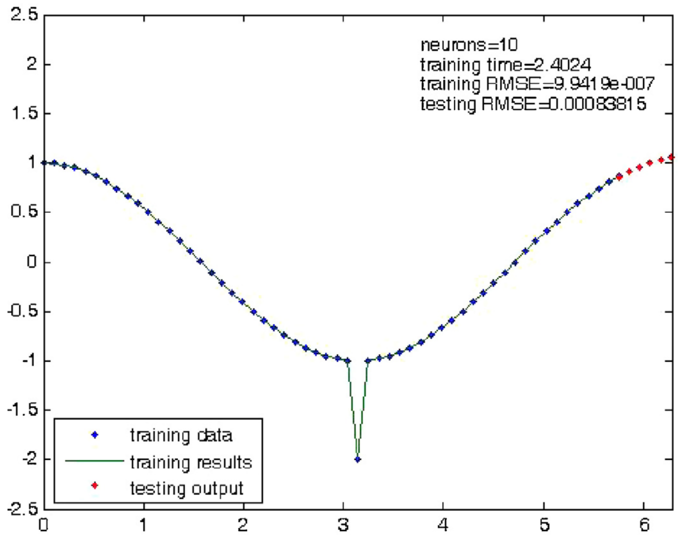

Fig. 5. Training and testing to dataset *B* with 10 neurons.

Theorem 2. ([43]) Let ϕ be bounded, measurable and sigmoidal. Then $\sum^{d}(\phi)$ is dense in $L^1([0,1]^d)$.

The following theorem is a generalization of the above results.

Theorem 3. ([44]) Let ϕ be a Lebesgue measurable function, not a.e. equal to a polynomial, satisfying $\int_a^b \left|\phi(x)\right|^p dx < \infty$ for all $a,\ \ b \in \mathbb{R}$. Let K be a compact set in $\mathbb{R}^d$. Then for any function $f \in L^p(K)\ \ (p \geq 1)$ and every $\varepsilon > 0$, there is a network $N \in \sum^{d}(\phi)$ such that

$\left\|N - f\right\|_{K,p} < \varepsilon$, where $\|g\|_{K,p} \equiv (\int_K \left|g(x)\right|^p dx)^{\frac{1}{p}}$.

For RBF neural networks we have the following results.

Theorem 4. ([35]) Let ϕ be a RBF function, Then for any function $f \in L^p(\mathbb{R})\ \ (p \geq 1)$ and every $\varepsilon > 0$, there is a network $N \in \sum^{d}(\phi)$ such that

$$\left\|N - f\right\|_p < \varepsilon,$$

where $\|g\|_p \equiv (\int_{\mathbb{R}} \left|g(x)\right|^p dx)^{\frac{1}{p}}$.

Theorem 5. ([32]) 1. Let $x \in [a,b]^k \subset \mathbb{R}^k$, and $f(x)$ be a multivariate continuous function, $x_i \in [a,b]^k$, $i = 0, 1, \cdots, n$ be an uniform grid partition to $[a,b]^k$, where $[a,b]$ is divided to s equal partition, and arrange by breadth-first such that $x_0, x_1, \cdots, x_n\ (n = s^k)$. Then

$$\left\|x_i - x_{i-1}\right\| = \frac{b-a}{n^{1/2k}} = \frac{b-a}{s^{1/2}}.$$

2. A depends on n, that is $A = A(n)$.

3. The real numbers f_i are the images of x_i under a multivariate continuous function $f(x)$, that is $f_i = f(x_i)$, $i = 0, 1, 2, \cdots, n$.

For each $\varepsilon > 0$, we can construct a decay RBF neural network $W_a(x, A(n))$, and there exists a function $A(n)$ and a natural number N such that, when $n > N$, we have

$\left|f(x) - W_a(x, n, A(n))\right| < \varepsilon$, for all $x \in [a,b]^k$.

The above theorems on approximation of continuous function can be carried out by many traditional machine learning systems with good generalization such as in Fig. 3.

4. Constructive multidimensional approximation of a function with one jump discontinuity point

In this section, we introduce a new constructive approach to reduce the overfitting phenomenon of machine learning system, especially neural networks, which can reduce the hidden neurons to optimal neural network architecture. The sample dataset with one noisy data is divided into two parts, the first part is considered to come from a simple continuous function. The other part is consisted of noisy data and is considered as one jumping discontinuous point of the continuous function. Then we can use many traditional methods to fit the continuous function with optimal architecture and good generalization, for the noisy part, we can construct a decay RBF neural network to fit it without influence the generalization to the first part and the whole machine learning system.

Definition 1. Consider $\psi(x)$ be a continuous real function, also the condition is given as $\lim_{x\to\infty}\psi(x) = 0 = o(e^{-x^2})$, and $\psi(0) \neq 0$. We call $\psi(x)$ is a decay RBF, and the decay RBF neural networks (DRNNs) can be written as

$$NW(x) = \sum_{j=0}^{n} c_j \psi(\lambda_j \left\|x - t_j\right\|), \quad x,\ t_j \in R^k, \tag{3}$$

where λ_j, t_j are inner weights, c_j outer weights respectively, and for $L^2(\mathbb{R})$, $\|x\| = (\int_R \left|x(t)\right|^2 dt)^{\frac{1}{2}}$

As we known that Gaussian function $\psi(x) = e^{-x^2}$ and wavelet functions, e.g. Mexican Hat wavelet $\psi(x) = (2/\sqrt{3})\pi^{-1/4}(1 - x^2)e^{-x^2/2}$ and Morlet wavelet $\psi(x) = (2/\sqrt{3})e^{-x^2/2}\cos 5x$ belong to decay RBF. The definition and nature of wavelet function are introduced in [45,46].

Replace $\varphi(x) = \psi(x)/\psi(0)$ to (3), we can get $\varphi(0) = 1$, $\lim_{x\to\infty}\varphi(x) = 0$ then we can rewriting (3) as follow

$$\begin{aligned} NW(x) &= \sum_{j=0}^{n} c_j \psi(\lambda_j \left\|x - t_j\right\|) = \sum_{j=0}^{n} c_j \psi(0) \frac{\psi(\lambda_j \left\|x - t_j\right\|)}{\psi(0)} \\ &= \sum_{j=0}^{n} k_j \varphi(\lambda_j \left\|x - t_j\right\|) \end{aligned}$$

Lemma 1. Consider $\varphi(x)$ is a decay RBF function, there exist real numbers k_1, k_2 and positive real number A, such that when $|x| > A$, we have $k_1 e^{-x^2} < \varphi(x) < k_2 e^{-x^2}$, and $\varphi(x)$ is bounded in $\mathbb{R}$.

Proof. As $\lim_{x\to\infty}\varphi(x) = 0 = o(e^{-x^2})$, then $\lim_{x\to\infty}\frac{\varphi(x)}{e^{-x^2}} = k$, and for each $\varepsilon > 0$ there exists a positive real number A, such that when $x > A$, we have $\left|\frac{\varphi(x)}{e^{-x^2}} - k\right| < \varepsilon$, that is, $-\varepsilon < \frac{\varphi(x)}{e^{-x^2}} - k < \varepsilon$, then $k - \varepsilon < \frac{\varphi(x)}{e^{-x^2}} < k + \varepsilon$ and

$$k_1 e^{-x^2} = (k - \varepsilon)e^{-x^2} < \varphi(x) < (k + \varepsilon)e^{-x^2} = k_2 e^{-x^2}.$$

Now, consider $y = f(x)\quad x \in \mathbb{R}^d$ is continuous except $x = x_0$, that is meaning, $\lim_{x\to x_0} f(x) \neq f(x_0)$, we can decompose $f(x)$ as two parts $f_c(x)$ and $f_d(x)$, it is shown as $f(x) = f_c(x), +f_d(x)$ where

$f_c(x) = \begin{cases} f(x) & x \neq x_0 \\ \lim_{x\to x_0} f(x) & x = x_0 \end{cases}$ is continuous and

$$f_d(x) = \begin{cases} 0 & x \neq x_0 \\ f(x_0) - \lim_{x\to x_0} f(x) = h_0 & x = x_0 \end{cases}$$

Example 1. For one-dimensional function:

$$y = f(x) = \begin{cases} \cos x & x \neq \pi \\ -2 & x = \pi \end{cases}$$

$f(x)$ can be decomposed as to $f_c(x)$ and $f_d(x)$, that is meaning $f(x) = f_c(x) + f_d(x)$, where

$f_c(x) = \cos x$ and $f_d(x) = \begin{cases} 0 & x \neq \pi \\ -1 & x = \pi \end{cases}$.

Example 2. 2-D function $z = f(x, y)$

$$z = \begin{cases} \dfrac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}} & x^2 + y^2 \neq 0 \\ 2 & x^2 + y^2 = 0 \end{cases}$$

We can decompose $f(x, y)$ to $f_c(x, y)$ and $f_d(x, y)$, so $f(x, y)$ can be denoted as

$$f(x, y) = f_c(x, y) + f_d(x, y),$$

where

$$f_c(x,y)=\begin{cases} f(x,y) & x^2+y^2\neq 0 \\ \lim\limits_{x^2+y^2\to 0} f(x,y)=1 & x^2+y^2=0 \end{cases}$$ is continuous and

$$f_d(x,y)=\begin{cases} 0 & x^2+y^2\neq 0 \\ f(0,0)-\lim\limits_{x^2+y^2\to 0} f(x,y)=1=h_0 & x^2+y^2=0 \end{cases}$$

Theorem 6. For each $\varepsilon>0$, there exists a constructive RBF neural networks $NW_d(x,A)$ and a positive real number A', such that when $A>A'$ we have
$\left\|f_d(x)-NW_d(x,A)\right\|<\varepsilon$.

Proof. Consider $NW_d(x,A)=h_0\varphi(A\|x-x_0\|)$ is a RBF neural networks with one neuron, so

$$NW_d(x_0,A)=h_0\varphi(A\|x-x_0\|)=h_0$$

$$\left\|f_d(x)-NW_d(x,A)\right\|=\left\|NW_d(x,A)\right\|$$
$$=(\int_R\left|h_0\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}$$
$$=\left|h_0\right|(\int_R\left|\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}$$
$$=\left|h_0\right|[(\int_{\|x-x_0\|<\delta}\left|\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}$$
$$+(\int_{\|x-x_0\|\geq\delta}\left|\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}]$$

By Lemma 1, $\left|\varphi(x)\right|<M>0$, then

$$\left|h_0\right|[(\int_{\|x-x_0\|<\delta}\left|\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}<\left|h_0\right|M\delta<\frac{\varepsilon}{2},$$

$$\textit{only if } \delta<\frac{\varepsilon}{2\left|h_0\right|M},$$

and when $A\|x-x_0\|>A_1$ that is, $A>\frac{A_1}{\|x-x_0\|}>\frac{A_1}{\delta}$, we can get

$$\left|h_0\right|(\int_{\|x-x_0\|\geq\delta}\left|\varphi(A\|x-x_0\|)\right|^2dx)^{\frac{1}{2}}$$
$$<\left|h_0\right|\int_R\left|k_2\right|e^{-A^2x^2}dx<\left|h_0\right|\left|k_2\right|\int_R e^{-A^2x^2}dx,$$
$$=\frac{\left|h_0\right|\left|k_2\right|}{A}\int_R e^{-x^2}dx=\frac{\left|h_0\right|\left|k_2\right|}{A}\sqrt{\pi}<\varepsilon$$

Then, when $A>\frac{|h_0||k_2|}{\varepsilon}\sqrt{\pi}$, we can get

$$\left\|f_d(x)-NW_d(x,A)\right\|<\frac{\varepsilon}{2}+\frac{\varepsilon}{2}=\varepsilon.$$

Remark 1. The decay neural network $NW_d(x,A)$ can be constructed to fit the noisy data without influence to the generalization of whole machine learning (Figs. 6–9).

Theorem 7. A function with one jumping discontinuity point can be repaired to a continuous function in $L^2(\mathbb{R})$ by decay RBF neural networks with each ε error.

Proof. It is obvious by Theorem 6.

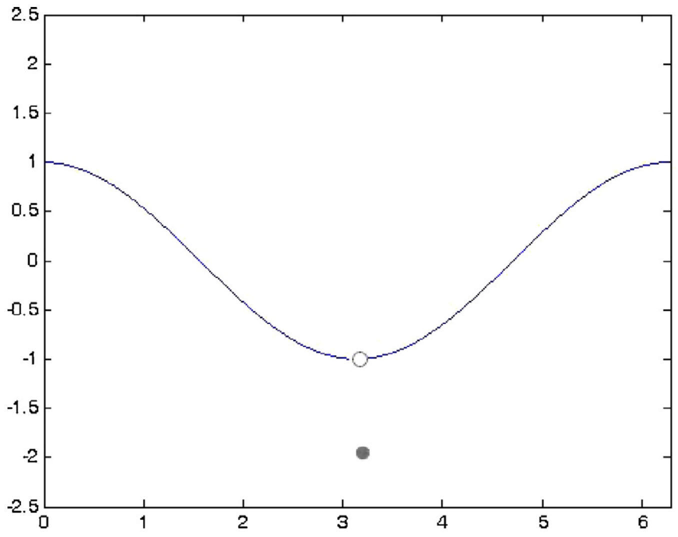

Fig. 6. The figure of function $y=f(x)$.

For Example 1, function (Fig. 6)

$$y=\begin{cases}\cos x & x\neq\pi \\ -2 & x=\pi\end{cases}$$

can be repaired by $\widetilde{y}=\cos x+e^{-A^2(x-\pi)^2}$ with Gaussian RBF function e^{-x^2} for each ε when $A>A_1$, having $\left\|y-\widetilde{y}\right\|<\varepsilon$ (Fig. 8)

Example 2, the function (Fig. 7)

$$z=\begin{cases}\dfrac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}} & x^2+y^2\neq 0 \\ 2 & x^2+y^2=0\end{cases}$$

can be repaired by $\widetilde{z}=f_c(x,y)+e^{-A^2(x^2+y^2)}$ with Gaussian RBF function $e^{-(x^2+y^2)}$ for each ε when $A>A_1$, having $\left\|z-\widetilde{z}\right\|<\varepsilon$ (Fig. 9)

Theorem 8. A function with one jumping discontinuity point can be approximated by a simplest neural networks with a decay RBF neural networks in $L^2(\mathbb{R})$ by each ε error.

Proof. It is obvious by Theorem 4 and Corollary 1.

Theorem 9. A function with one jumping discontinuity point $y=f(x)$, $x\in E\subset\mathbb{R}^d$ can be constructively fitted by a decay RBF neural networks in $L^2(\mathbb{R}^d)$ by each $\varepsilon>0$ error.

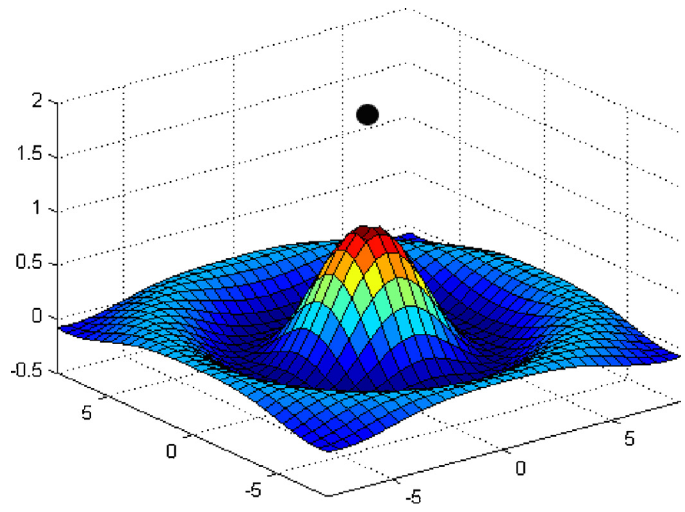

Fig. 7. The figure of function $z=f(x,y)$.

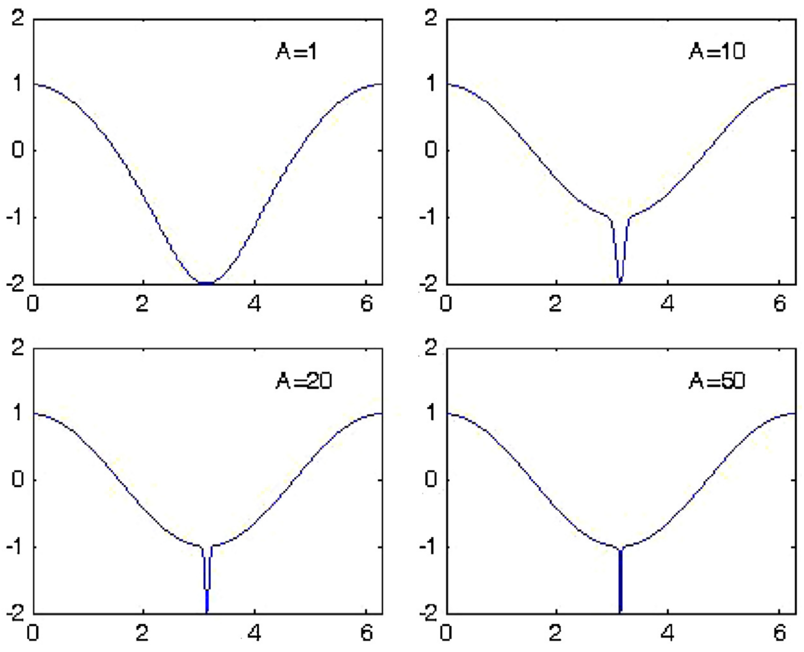

Fig. 8. The repaired approximation of function $y = f(x)$ with decay RBF neural networks.

Proof. As $f(\mathrm{x}) = f_c(\mathrm{x}) + f_d(\mathrm{x})$, $f_c(\mathrm{x}) \in E \subset C(\mathbb{R}^d)$, from Theorem 4, for each $\varepsilon > 0$, we can construct a decay RBF neural network $NW_c(\mathrm{x}, A)$, and there exists a natural number A_1 such that, when $A > A_1$, we have

$\left|f_c(\mathrm{x}) - NW_c(\mathrm{x}, A)\right| < \frac{\varepsilon}{2}$, for all $\mathrm{x} \in E \subset \mathbb{R}^d$.

And from Theorem 6 for each $\varepsilon > 0$, there exist a constructive RBF neural networks $NW_d(x, A)$ and a positive real number A_2, such that when $A > A_2$ we have

$$\left|\left|f_d(x) - NW_d(x, A)\right|\right| < \frac{\varepsilon}{2}.$$

Then when $A > A_0 = \max\left\{A_1, A_2\right\}$ we have

$$\left|f(x) - (NW_c(x, A) + NW_d(x, A))\right|$$

$$\leq \left|f_c(x) - NW_c(x, A)\right| + \left|f_d(x) - NW_d(x, A)\right| \tag{4}$$

$$< \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$$

And $NW(x, A) = NW_c(x, A) + NW_d(x, A)$ is also a constructive RBF neural networks.

Based on the above theorems, a sample dataset with one noisy point can be divided into two parts; the first part can be considered

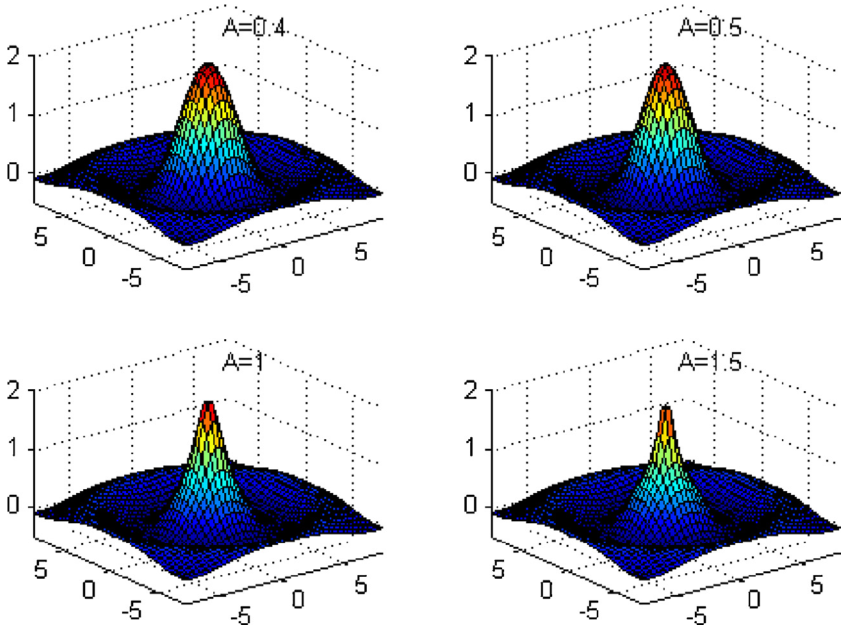

Fig. 9. The repaired approximation of the function $z = f(x, y)$ with decay RBF neural networks.

from a simple continuous function which can be fit by traditional machine learning system such as Fig. 1, the noisy part can be fit by a decay neural network with constructive approach. Then the whole machine learning system has optimal architecture and well generalization of part one, which can reduced the overfitting phenomenon of machine learning system.

5. Constructive multidimensional approximation of function with finite number of jump discontinuity point

The content of this section is the extension of Section IV to the situation of m noisy points which has the same conclusion of optimizing neural network architecture and generalization.

Let $y=f(\mathrm{x})\quad \mathrm{x}\in\mathbb{R}^d$ has m jumping discontinuity points $\mathrm{x}_j,\quad j=1,2,\cdots,m$, that is, $\lim\limits_{\mathrm{x}\to\mathrm{x}_j} f(\mathrm{x})\neq f(\mathrm{x}_j)\ j=1,2,\cdots,m$. We can decompose $f(\mathrm{x})$ to $f_c(\mathrm{x})$ and $f_d(\mathrm{x})$, that is, $f(\mathrm{x})=f_c(\mathrm{x})+f_d(\mathrm{x})$, where

$$f_c(x)=\begin{cases} f(x) & x\neq x_j \quad j=1,2,\cdots,m \\ \lim\limits_{x\to x_j} f(x) & x=x_j \quad j=1,2,\cdots,m \end{cases} \tag{5}$$

is continuous and

$$f_d(x)=\begin{cases} 0 & x\neq x_j \quad j=1,2,\cdots,m \\ f(x_j)-\lim\limits_{x\to x_j} f(x)=h_j & x=x_j \quad j=1,2,\cdots,m \end{cases} \tag{6}$$

Theorem 10. If $y=f(\mathrm{x})\quad \mathrm{x}\in\mathbb{R}^d$ has m jumping discontinuity points $\mathrm{x}_j\quad j=1,2,\cdots,m$, that is, $\lim\limits_{\mathrm{x}\to\mathrm{x}_j} f(\mathrm{x})\neq f(\mathrm{x}_j)\quad j=1,2,\cdots,m$

and $f(\mathrm{x})=f_c(\mathrm{x})+f_d(\mathrm{x})$ such as (5) and (6), for each $\varepsilon>0$, there exist a constructive RBF neural networks $NW_d(\mathrm{x},A)$ and a positive real number A', such that when $A>A'$ we have

$$\left\|f_d(\mathrm{x})-NW_d(\mathrm{x},A)\right\|<\varepsilon$$

Proof. Firstly let $f_d(x)=\sum_{j=1}^{m} f_{dj}(x)$, where

$$f_{dj}(x)=\begin{cases} 0 & x\neq x_j \\ f(x_j)-\lim\limits_{x\to x_j} f(x)=h_j & x=x_j \end{cases}.$$

Then for each jumping discontinuity point x_j, by theorem 6, for each $\varepsilon>0$, there exist a constructive RBF neural networks $NW_{dj}(\mathrm{x},A)=h_j\varphi(A\left\|\mathrm{x}-\mathrm{x}_j\right\|)$ with one neuron and a positive real number A'_j, such that when $A>A'_j$ we have

$$\left\|f_{dj}(x)-NW_{dj}(x,A)\right\|<\frac{\varepsilon}{m}.$$

Then, we construct

$$NW_d(x,A)=\sum_{j=1}^{m} NW_{dj}(x,A)=\sum_{j=1}^{m} h_j\varphi(A\left\|x-x_j\right\|),\text{when}$$

$A>A'=\max\left\{A'_1,A'_2,\cdots,A'_m\right\}$, we have

$$\left\|f_d(x)-NW_d(x,A)\right\|\leq\sum_{j=1}^{m}\left\|f_{dj}(x)-NW_{dj}(x,A)\right\|$$

$$<m\frac{\varepsilon}{m}=\varepsilon$$

Theorem 11. A function with m jumping discontinuity points can be repaired to a continuous function in $L^2(\mathbb{R})$ by decay RBF neural networks with each ε error.

Proof. It is obvious by Theorem 10.

Theorem 12. A function with m jumping discontinuity point can be approximated by a simplest neural networks and a decay RBF neural networks in $L^2(\mathbb{R})$ by each ε error.

Proof. It can be proved similarly to Theorem 8.

Theorem 13. A function with m jumping discontinuity points $y=f(\mathrm{x}),\quad \mathrm{x}\in E\subset\mathbb{R}^d$ can be constructively approximated by a decay RBF neural networks in $L^2(\mathbb{R}^d)$ by each $\varepsilon>0$ error.

Proof. It can be proved similarly to Theorem 9.

6. A real world problem of stock data

In this section, a practical problem of stock data is presented to verify above theory. All stock market trends are fast changing. It is affected by not only the individual investors and many institutional investors, but also impacted by domestic political, economic situations and many other factors. Therefore, it is very difficult to build a classical parameter model to predict the market movement [47]. But it is easy to build a NNs model to fit the stock dataset to predict the stock closing price.

We first collected the sample data of closing price of Chongqing Iron & Steel (601005) in Chinese Shanghai stock market from internet stock database, which is called as dataset C. The collection period is from 4 January 2012 to 8 October 2012 and the number of data totaled 120 (Fig. 10).

From Fig. 10, we can observe there is a noise in $x=35$ (2012.6.1), which have too much information to move it. And we can create a mathematical model to detect the noisy data as follows:

If $\left|\frac{f(x_i)-f(x_{i-1})}{f(x_{i-1})}\right|>k>0,\ \left|\frac{f(x_i)-f(x_{i+1})}{f(x_{i+1})}\right|>k>0$ and $(f(x_i)-f(x_{i-1}))(f(x_i)-f(x_{i+1}))>0$, where k is a control parameter based on specific problems, we consider $f(x_i)$ as a noisy point. In Fig. 10, we take $k=0.08$. Then we can define the follow detection function:

$$d(x_i)=\begin{cases} 1, & \left|\dfrac{f(x_i)-f(x_{i-1})}{f(x_{i-1})}\right|>k>0,\quad \left|\dfrac{f(x_i)-f(x_{i+1})}{f(x_{i+1})}\right|>k>0 \quad\text{and}\quad (f(x_i)-f(x_{i-1}))(f(x_i)-f(x_{i+1}))>0 \\ 0, & \text{otherwise} \end{cases}$$

So in the noisy point x_i, $d(x_i)=1$, otherwise, $d(x_i)=0$. We applied the detection function to Fig. 10, and get the results in Fig. 11.

From the result of Fig. 11, we can see $f(x_{35})$ is a noisy point. In the following experiments, we will use 115 points of the dataset to train the NNs and the other 5 points to test the NNs.

In Fig. 12, the NNs with 6 hidden neurons fit the dataset C with 17.4565 CPU training time, 0.0016164 training RMSE and 0.0013742 testing RMSE.

In order to fit dataset C more well, if we increase the hidden neurons to 9 neurons, we get the results as Fig. 13. Although the training RSME decreased to 0.0014821, but the testing RMSE increased to

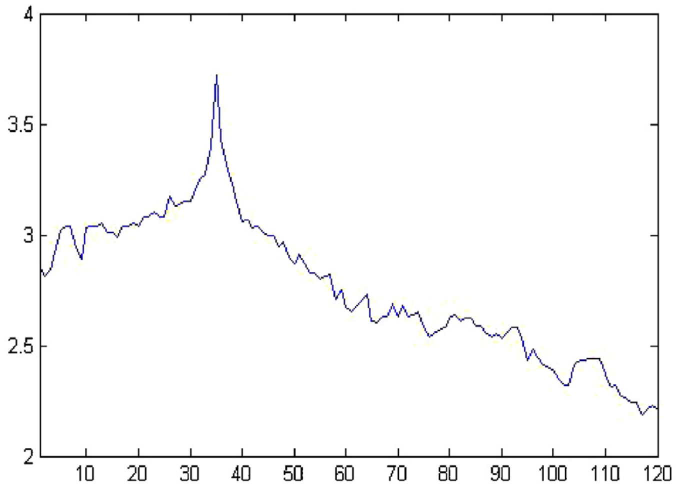
Fig. 10. Closing price of Chongqing Iron & Steel in 120 days.

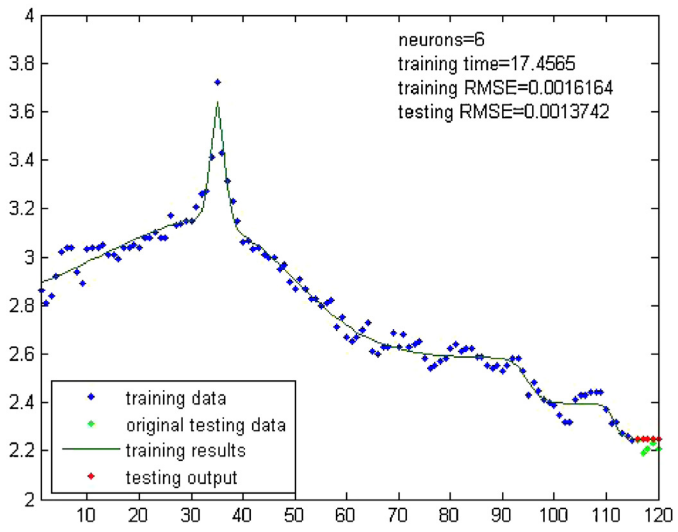

Fig. 12. Training and testing to dataset *C* with 6 neurons.

0.017929 that is larger 13.05 times than that in Fig. 12 because of the noisy point overfitting.

Now, we consider the dataset *C* comes from a function $y=f(x)$ with a jumping discontinuous point at $x=35$. Then we decompose function $f(x)$ as two parts $f_c(x)$ and $f_d(x)$, so, $f(x)=f_c(x)+f_d(x)$, where

$$f_c(x)=\begin{cases} f(x) & x\neq 35 \\ \lim_{x\to 35} f(x)=\frac{1}{2}(f(34)+f(36))=3.42 & x=35 \end{cases}$$

is continuous part, which dataset *D* comes from, and

$$f_d(x)=\begin{cases} 0 & x\neq 35 \\ 3.72-3.42=0.3 & x=35 \end{cases}.$$

For the dataset *D* which came from a smoother function $f_c(x)$, a BP NNs with 6 hidden neurons can fit it very well in Fig. 14 with 0.0012357 training RMSE and 0.00083884 testing RMSE that is smaller 21.37 times than that in Fig. 13 and 1.64 times that in Fig. 12 because of no overfitting the noisy point with less hidden neurons.

Then we constructed a constructive RBF NNs $NW_d(x, A)$ with $A=50$ to approximate the function $f_d(x)$, and the comprehensive results are presented in Fig. 15, which have less hidden neurons and smaller training and testing RMSE that verified the correctness of above theory.

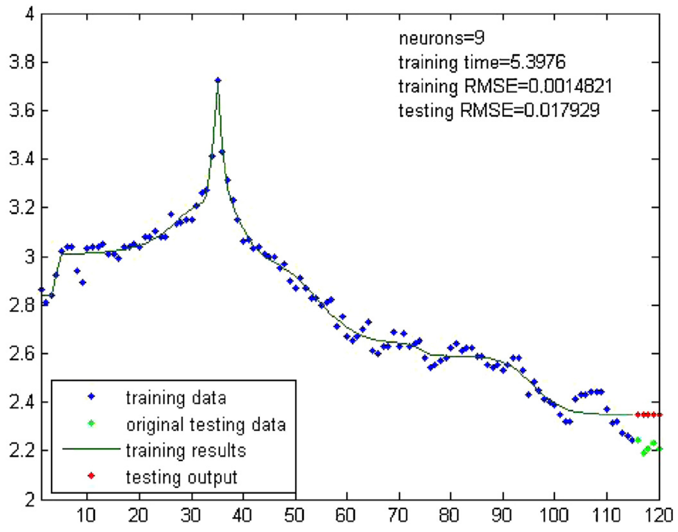

Fig. 13. Training and testing to dataset *C* with 9 neurons.

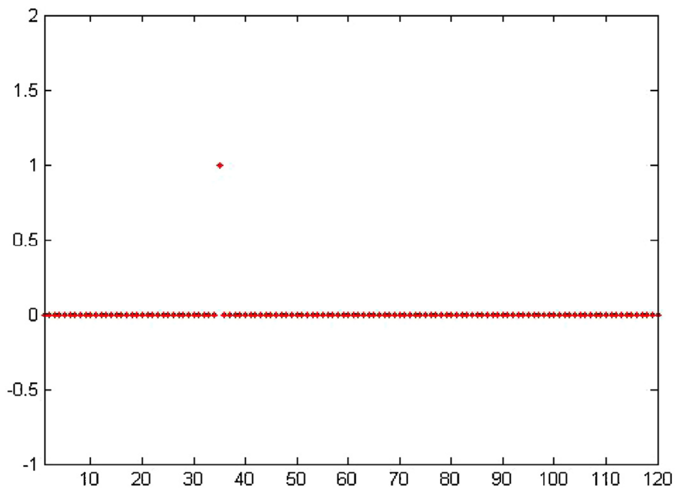
Fig. 11. The result of noisy detection.

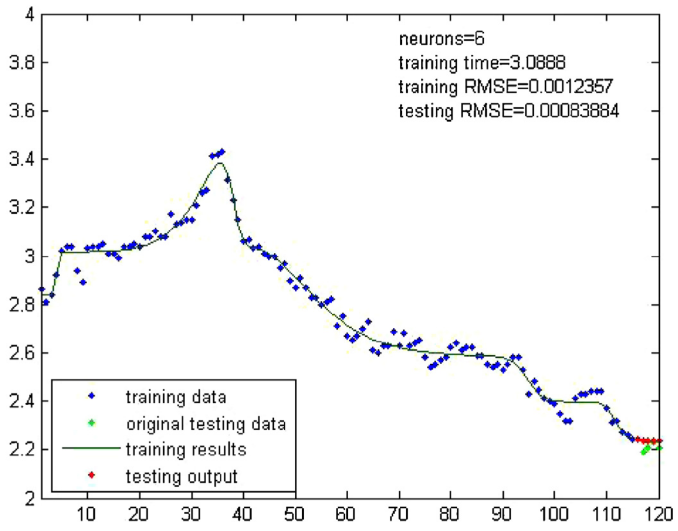

Fig. 14. Training and testing to dataset *D* with 6 neurons.

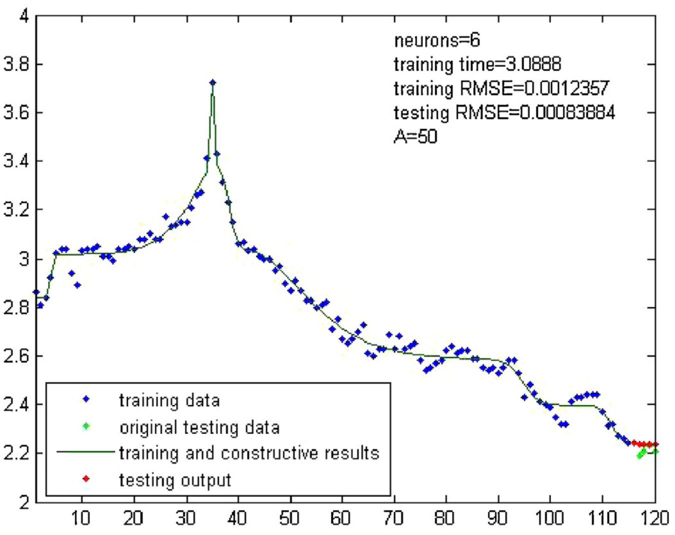

Fig. 15. The comprehensive results with dataset *C*.

Remark 1. In real practical problems, because of many factors, noise data always exist. And we always assume that noise data can be distinguished from non-noise one based on some practical principle or experiences such as above example. These noise data can make the mathematical model such as neural networks very complex or overfitting. So the traditional methods of processing noise data is to remove them before approximation through various algorithms such as wavelet transform [15–18], principal component analysis [19,20] and various filtering algorithms [21–24]. But sometimes, the "noisy" data we think of ways to remove are often some singular values of a process which contains important information [25] which have other uses in the next step and would not be removed.

Remark 2. In above real stock price example, based on the practical experience, the data point at $x = 35$ changes so much bigger than others that makes the neural network for prediction overfitting. So we consider this data is a noise data, but we does not remove it in the prediction process.

7. Conclusions

In this paper, after analyzing the reasons of poor generalization and overfitting in neural networks, we consider some noise data as a singular values of a continuous function - jump discontinuity point. The continuous part can be approximated with the simplest neural networks, which have good generalization performance and optimal network architecture, by traditional algorithms such as constructive algorithm for feed-forward neural networks with incremental training, BP algorithm, ELM algorithm, various constructive algorithm, RBF approximation and SVM. At the same time, we will construct RBF neural networks to fit the singular value with every ε error in $L^2(\mathbb{R}^d)$, and we prove that a function with m jumping discontinuity points can be approximated by the simplest neural networks with a decay RBF neural networks in $L^2(\mathbb{R})$ by each ε error, and a function with m jumping discontinuity point $y = f(x), \quad x \in E \subset \mathbb{R}^d$ can be constructively approximated by a decay RBF neural networks in $L^2(\mathbb{R}^d)$ by each $\varepsilon > 0$ error and the constructive part have no generalization influence to the whole machine learning system which will optimize neural network architecture and generalization performance, reduce the overfitting phenomenon by avoid fitting the noisy data. And a real world problem about stock closing price with jumping discontinuity have been presented and verified the correctness of the theory.

Acknowledgments

This work was supported by the World Class University (WCU, R32-2013-000-20014-0) and Basic Science Research Program 20100020942, MEST 2012-002521, NRF, Korea.

References

[1] J.R. Noriega, H. Wang, A direct adaptive neural-network control for unknown nonlinear systems and its application, IEEE Transactions on Neural Networks 9 (1998) 27–34.
[2] G.A. Carpenter, S. Grossberg, The ART of adaptive pattern recognition by a self-organizing neural network, Computer 21 (1988) 77–88.
[3] Y.-J. Liu, C. Chen, G.-X. Wen, S. Tong, Adaptive neural output feedback tracking control for a class of uncertain discrete-time nonlinear systems, IEEE Transactions on Neural Networks 22 (2011) 1162–1167.
[4] S. Ge, C. Hang, T. Zhang, Adaptive neural network control of nonlinear systems by state and output feedback, IEEE Transactions on Systems, Man, and Cybernetics, Part B 29 (1999) 818–828.
[5] M.M. Polycarpou, Stable adaptive neural control scheme for nonlinear systems, IEEE Transactions on Automatic Control 41 (1996) 447–451.
[6] K. Hagiwara, K. Fukumizu, Relation between weight size and degree of over-fitting in neural network regression, Neural Networks 21 (2008) 48–58.
[7] I.V. Tetko, D.J. Livingstone, A.I. Luik, Neural network studies. 1. Comparison of overfitting and overtraining, Journal of Chemical Information and Computer Sciences 35 (1995) 826–833.
[8] C. Schittenkopf, G. Deco, W. Brauer, Two strategies to avoid overfitting in feedforward networks, Neural Networks 10 (1997) 505–516.
[9] Z. Yu, S. Song, G. Duan, R. Pei, W. Chu, The Design of RBF Neural Networks for Solving Overfitting Problem. WCICA 2006 The Sixth World Congress on Intelligent Control and Automation, vol. 1, IEEE, 2006, pp. 2752–2756.
[10] M.P. Perrone, L.N. Cooper, When networks disagree: ensemble methods for hybrid neural networks, 1992 (DTIC document).
[11] F. Tay, L. Cao, Application of Support Vector Machines in Financial Time Series Forecasting, vol. 29, Omega-Oxford-Pergamon Press, 2001, pp. 309–317.
[12] C. Doan, S. Liong, Generalization for multilayer neural network: Bayesian regularization or early stopping, 2004.
[13] J. Liu, O. Demirci, V.D. Calhoun, A parallel independent component analysis approach to investigate genomic influence on brain function, Signal Processing Letters, IEEE 15 (2008) 413–416.
[14] S. Soltani, On the use of the wavelet decomposition for time series prediction, Neurocomputing 48 (2002) 267–277.
[15] G.K. Prasad, J. Sahambi, Classification of ECG arrhythmias using multi-resolution analysis and neural networks, IEEE 1 (2003) 227–231.
[16] L.Y. Shyu, Y.H. Wu, W. Hu, Using wavelet transform and fuzzy neural network for VPC detection from the Holter ECG, IEEE Transactions on Biomedical Engineering 51 (2004) 1269–1273.
[17] Y. He, Y. Tan, Y. Sun, Wavelet Neural Network Approach for Fault Diagnosis of Analogue Circuits, vol. 151, IET, 2004, pp. 379–384.
[18] B. Chen, X. Wang, S. Yang, C. McGreavy, Application of wavelets and neural networks to diagnostic system development, 1, Feature extraction, Computers and Chemical Engineering 23 (1999) 899–906.
[19] J. Karhunen, E. Oja, L. Wang, R. Vigario, J. Joutsensalo, A class of neural networks for independent component analysis, IEEE Transactions on Neural Networks 8 (1997) 486–504.
[20] A. Hyvärinen, E. Oja, Independent component analysis: algorithms and applications, Neural Networks 13 (2000) 411–430.
[21] S. Shah, F. Palmieri, M. Datum, Optimal filtering algorithms for fast learning in feedforward neural networks, Neural Networks 5 (1992) 779–787.
[22] J.T. Connor, R.D. Martin, L. Atlas, Recurrent neural networks and robust time series prediction, IEEE Transactions on Neural Networks 5 (1994) 240–254.
[23] R. Feraund, O.J. Bernier, J.E. Viallet, M. Collobert, A fast and accurate face detector based on neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 23 (2001) 42–53.
[24] S.S. Haykin, Neural Networks and Learning Machines, vol. 3, Prentice Hall, 2009.
[25] L. Fan, S. Wang, H. Wang, T. Guo, Singular points detection based on zero-pole model in fingerprint images, IEEE Transactions on Pattern Analysis and Machine Intelligence 30 (2008) 929–940.
[26] M.M. Islam, X. Yao, K. Murase, A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks 14 (2003) 820–834.
[27] D. Liu, T.S. Chang, Y. Zhang, A constructive algorithm for feedforward neural networks with incremental training, IEEE Transactions On Circuits and Systems Part 1 Fundamental Theory and Applications 49 (2002) 1876–1879.
[28] S. Haykin, N. Network, A comprehensive foundation, Neural Networks 2 (2004).
[29] H. Simon, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999.
[30] G.B. Huang, Q.Y. Zhu, C.K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, IEEE 2 (2004) 985–990.
[31] G.B. Huang, L. Chen, Convex incremental extreme learning machine, Neurocomputing 70 (2007) 3056–3062.

[32] M. Hou, X. Han, Constructive approximation to multivariate function by decay RBF neural network, IEEE Transactions on Neural Networks 21 (2010) 1517–1523.
[33] B. Llanas, F. Sainz, Constructive approximate interpolation by neural networks, Journal of Computational and Applied Mathematics 188 (2006) 283–308.
[34] M.M. Islam, M.A. Sattar, M.F. Amin, X. Yao, K. Murase, A new constructive algorithm for architectural and functional adaptation of artificial neural networks, IEEE Transactions on Systems, Man, and Cybernetics, Part B 39 (2009) 1590–1605.
[35] J. Park, I.W. Sandberg, Universal approximation using radial-basis-function networks, Neural Computation 3 (1991) 246–257.
[36] M.J. Er, S. Wu, J. Lu, H.L. Toh, Face recognition with radial basis function (RBF) neural networks, IEEE Transactions on Neural Networks 13 (2002) 697–710.
[37] N. Mai-Duy, T. Tran-Cong, Approximation of function and its derivatives using radial basis function networks, Applied Mathematical Modelling 27 (2003) 197–220.
[38] C.C. Chang, C.J. Lin, LIBSVM: a library for support vector machines, ACM Transactions on Intelligent Systems and Technology (TIST) 2 (2011) 27.
[39] W. Kai, Y., Jufeng, S., Guangshun, W. Qingren, An Expanded Training Set Based Validation Method to Avoid Overfitting for Neural Network Classifier, ICNC '08, Fourth International Conference on Natural Computation, Vol. 3, 2008, pp. 83–87.
[40] Z.P. Liu, J.P. Castagna Avoiding Overfitting Caused by Noise Using a Uniform Training Mode, IJCNN '99, International Joint Conference on Neural Networks, Vol. 3, 1999. pp. 1788–1793.
[41] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta Numerica 8 (1999) 143–195.
[42] J.I. Mulero-Martínez, Best approximation of Gaussian neural networks with nodes uniformly spaced, IEEE Transactions on Neural Networks 19 (2008) 284–298.
[43] G. Cybenko, Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals, and Systems (MCSS) 2 (1989) 303–314.
[44] R.M. Burton, H.G. Dehling, Universal approximation in p-mean by neural networks, Neural Networks 11 (1998) 661–667.
[45] C.K. Chui, An introduction to wavelets, Vol. 1, Academic Press, 1992.
[46] B. Delyon, A. Juditsky, A. Benveniste, Accuracy analysis for wavelet approximations, IEEE Transactions on Neural Networks 6 (1995) 332–348.
[47] C.J. Huang, P.W. Chen, W.T. Pan, Using multi-stage data mining technique to build forecast model for Taiwan stocks, Neural Computing and Applications (2011) 1–7.