Charted Depth Interpolation: Neuron Network Approaches

Shi Chaojian
Shanghai Maritime University
1550 Pudong Dadao, Shanghai 200135, P. R. China
cjshi@shmtu.edu.cn

ABSTRACT: Continuous depth data are often required in applications of both onboard systems and maritime simulation. But data available are usually discrete and irregularly distributed. Based on the neuron network technique, methods of interpolation to the charted depth are suggested in this paper. Two algorithms based on Levenberg-Marquardt back-propaganda and radial-basis function networks are investigated respectively. A dynamic neuron network system is developed which satisfies both real time and mass processing applications. Using hyperbolic paraboloid and typical chart area, effectiveness of the algorithms is tested and error analysis presented. Special process in practical applications such as partition of lager areas, normalization and selection of depth contour data are also illustrated.

KEY WORDS: charted depth, neuron network, function approximation, spatial interpolation

1 Introduction

In practical applications such as geometric information systems (Wu 2002), maritime simulation, ship automation and harbor and waterway design (Shi 2003 and Xiao 2002), etc, we usually need continuous water depth data. Because of restricted observation measures and limited information resources, the depth data available are usually discrete and irregularly distributed, such as those either on paper chart or in ECDIS database (IHO 1996). A systematic approach of interpolation is required for practical purposes. This kind of problem, known as spatial interpolation, can be addressed using various methods, which include inverse distance weighting, interpolating polynomials, splines, power and Fourier series fitting and others (Cressie 1993). Some of the methods may lack the desired accuracy and some are hard to implement in tow dimensional environment. More complicated methods like kriging perform well but also need much effort and computer time (Zimmerman 1999). Neuron networks behave very well in nonlinear function approximation and are worth being investigated.

2 Neuron Network Approaches

2.1 Function Approximation

Let the surface of the sea bottom be denoted by

$$z = f(x,y)$$

where x and y are horizontal coordinates and z is the water depth. Employing the superior performance of neuron network for approximating nonlinear functions, $f(x,y)$ can be effectively evaluated by a properly designed network architecture.

The universal approximation theorem for a nonlinear input-output mapping can be used to solve the problem: Let $\varphi(\ .\ )$ be a nonconstant, bounded, and monotone increasing continuous function. Let I_{m_0} denote m_0-dimentional unit hypercube $[0,1]^{m_0}$. The space of continuous functions on I_{m_0} is denoted by $C(I_{m_0})$. Then,

given any function $f \in C(I_{m_0})$ and ε >0, there exist a integer m_1 and sets of real constants α_i, b_i, and w_{ij}, where i=1,2,...m_1 and j=1,2,...m_0, such that we may define

$$F(x_1, x_2, ...x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi(\sum_{j=1}^{m_0} w_{ij} x_j + b_i)$$

as an approximate realization of the function $f(\ .\ )$; that is,

$$\left| F(x_1, x_2, ...x_{m_0}) - f(x_1, x_2, ...x_{m_0}) \right| < \varepsilon$$

for all $x_1, x_2, ...x_{m_0}$ that lie in the input space.

It can be accepted that the surface of the sea bottom meets the requirement as a continuous function and the normalized surface function $f \in C(I_{m_0})$. Selecting *logsig* as the transfer functions of the neurons meets the need to be nonconstant, bounded and monotone increasing continuous. To approximate the sea bottom surface, therefore, we can employ a network architecture as shown in Fig 1. The transfer function of hidden layer is *logsig*, and that of output layer, *purelin*. The free parameter vector of the network is

$$\mathbf{x}^T = [x_1, x_2, ...x_n] = [w^1_{1,1}, w^1_{1,2}, ...w^1_{S^1,R}, b^1_1, ...b^1_{S^1}, w^2_1, ...w^2_{S^1}, b^2_1]$$

The charted depth data in a proper area can be used as training set. The network is trained by backpropagation algorithm and the free parameter vector is modified to approximate effectively the surface function, $f(x,y)$, of the sea bottom.

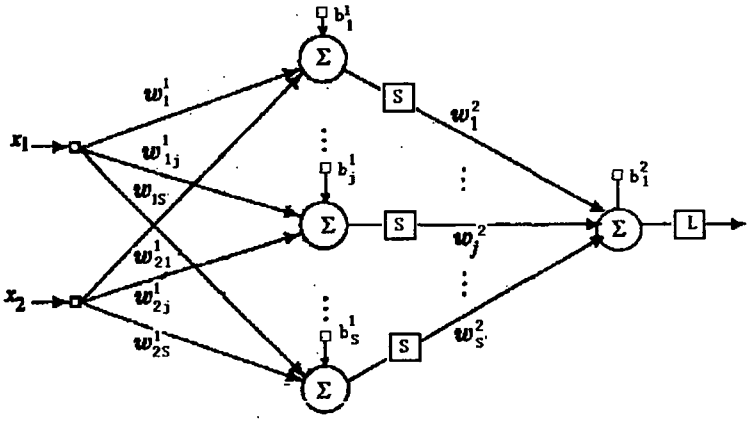

Fig.1 Fuction approximation BP network

The standard backpropagation algorithm is popular and easy to implement. It's converging speed is slow, however. The heuristic modifications such as the use of momentum or variable learning rate show some improvements but hardly are satisfactory in practical applications. Levenberg-Marquardt backpropagation (LMBP) proves to be an effective algorithm. It appears to be the fastest neuron network training algorithm for moderate number of network parameters (Hagan 1996).

2.2 Function Interpolation

The performance of the backpropagation network approximation generally meets the requirements of practical application. But the approximation surface usually does not pass through the observed data, which are used as training data points, and there exist errors over the known data points. In some special applications, which require accurate value over observed data points, radial-basis function (RBF) network is preferred.

The interpolation problem may be stated: given a set of N different points $\{ \mathbf{x}_i \in \Re^{m_0} \mid i = 1,2,...N \}$ and a corresponding set of N real numbers $\{ d_i \in \Re^1 \mid i = 1,2,...N \}$, find a function $F : \Re^{m_0} \to \Re^1$ that satisfies the interpolation condition:

$$F(\mathbf{x}_i)=d_i, \qquad i=1,2,\ldots N$$

Use the radial-basis function technique and choose a function F that has the following form:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(\| \mathbf{x} - \mathbf{x}_i \|) \qquad (1)$$

where $\{\varphi(\| \mathbf{x} - \mathbf{x}_i \|) \mid i = 1,2,\ldots N\}$ is a set of N nonlinear functions, known as radial-basis functions, and $\| \cdot \|$ denotes a norm. The known data points $\mathbf{x}_i \in \Re^{m_0} \mid i = 1,2,\ldots N$, are taken as center of the radial-basis functions.

From (1) and (2), we can write:

$$\mathbf{\Phi w = d}$$

where

$$\mathbf{\Phi} = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{bmatrix} \quad (\varphi_{ij} = \varphi(\| \mathbf{x}_j - \mathbf{x}_i \|) \mid i, j = 1,2,\ldots N)$$

$$\mathbf{d}^T = [d_1, d_2, \ldots d_N],$$
$$\mathbf{w}^T = [w_1, w_2, \ldots w_N] \qquad (2)$$

By Micchelli theorem , Let $\mathbf{x}_i \in \Re^{m_0} \mid i = 1,2,\ldots N$ be a set of distinct points. Then the N-by-N interpolation matrix $\mathbf{\Phi}$, whose ji-th elements is $\varphi_{ij}=\varphi(r_{ij})=\varphi(\| \mathbf{x}_j-\mathbf{x}_i \|)$, is nonsingular (Micchelli 1986). Therefore we can write,

$$\mathbf{w}=\mathbf{\Phi}^{-1}\mathbf{d} \qquad (3)$$

Many radial-basis functions are covered by Micchelli theorem. The following ones are commonly applied (Haykin 1999):

$$\varphi(r) = (r^2 - c^2)^{1/2}$$
$$\varphi(r) = (r^2 - c^2)^{-1/2}$$
$$\varphi(r) = \exp(-\frac{r^2}{2\sigma^2})$$

where c, σ are constants.

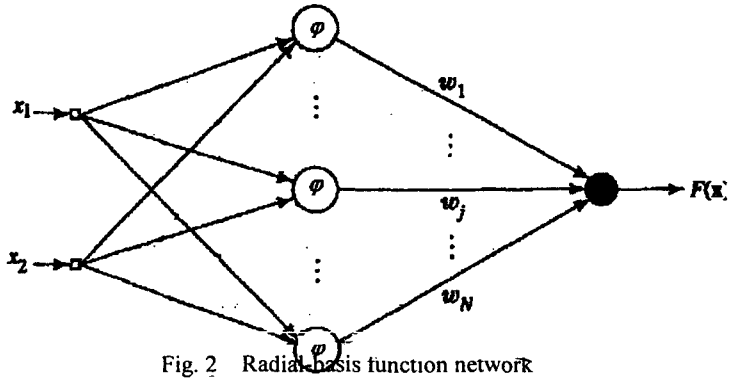

Fig. 2 Radial-basis function network

The architecture of radial-basis function network for the function interpolation is shown in Fig.2. The number of neurons on the hidden layer is identical to the number of data in the training set, N. The w in (2) is the free parameter vector. The output of the network is (1) and the free parameters are determined by (3).

3 Tests and Implementation Analysis

3.1 Curve Function Test

In order to verify the performance of the network for interpolation, Tests have been performed on a hyperbolic paraboloid,

$$z = (\frac{x}{0.5})^2 - (\frac{y}{0.6})^2$$

The result of the interpolation is shown in Fig 3. The surface is generated by the radial-basis function network. The dark points represent the training data.

Table 1 Interpolated values from the tested function

x	y	z	RBF		LMBP (16)		LMBP (20)	
			Output	Error	Output	Error	Output	Error
-0.199	0.227	**0.015**	0.049	*0.034*	0.009	*-0.006*	0.017	*0.003*
-0.133	-0.234	**-0.082**	-0.056	*0.025*	-0.090	*-0.009*	-0.079	*0.003*
0.012	0.418	**-0.485**	-0.460	*0.025*	-0.481	*0.004*	-0.485	*0.000*
-0.630	0.667	**0.350**	0.402	*0.052*	0.288	*-0.062*	0.339	*-0.011*
0.292	-0.249	**0.170**	0.176	*0.006*	0.191	*0.021*	0.179	*0.009*
-0.304	0.527	**-0.403**	-0.364	*0.040*	-0.438	*-0.034*	-0.401	*0.002*
-0.108	0.820	**-1.822**	-1.792	*0.030*	-1.818	*0.004*	-1.827	*-0.004*
-0.051	-0.051	**0.003**	0.028	*0.025*	0.002	*-0.001*	0.008	*0.005*
0.704	0.608	**0.956**	0.976	*0.020*	0.928	*-0.028*	0.961	*0.005*
0.446	0.399	**0.353**	0.367	*0.014*	0.380	*0.027*	0.353	*-0.000*
-0.438	-0.712	**-0.641**	-0.609	*0.033*	-0.631	*0.010*	-0.645	*-0.004*
0.512	-0.610	**0.015**	0.008	*-0.007*	0.070	*0.055*	0.020	*0.005*
-0.153	-0.953	**-2.430**	-2.426	*0.005*	-2.428	*0.002*	-2.440	*-0.010*
0.163	-0.928	**-2.286**	-2.294	*-0.008*	-2.267	*0.020*	-2.287	*0.000*
-0.462	-0.687	**-0.459**	-0.426	*0.034*	-0.451	*0.009*	-0.466	*-0.007*
-0.891	0.406	**2.714**	2.779	*0.065*	2.685	*-0.029*	2.719	*0.004*
-0.382	-0.297	**0.339**	0.375	*0.036*	0.318	*-0.021*	0.337	*-0.002*
-0.602	-0.011	**1.450**	1.501	*0.051*	1.425	*-0.025*	1.442	*-0.008*
-0.156	-0.171	**0.016**	0.044	*0.028*	0.003	*-0.014*	0.018	*0.001*
-0.961	0.312	**3.425**	3.497	*0.072*	3.412	*-0.012*	3.421	*-0.004*

From $\{(x, y) \mid -1<x<1, -1<y<1\}$, we choose 20 data points randomly and get the interpolated value from the trained RBF and LMBP networks respectively. The results are shown in Table 1. For LMBP results, the numbers in the parentheses denote the number of neurons in the hidden layer. Table 2 shows the statistical result from values of 20 000 random data points. The consequence shows that either RBF or LMBP network demonstrates a good performance and meets the needs in most practical applications.

Table 2 Error of interpolated values

	RBF	LMBP (16)	LMBP (20)
Average	0.025	-0.013	-0.001
Variance	0.057	0.064	0.020

3.2 Chart Area Test

Tests are also made in the chart area shown in Fig. 4. The training and interpolation are implemented within the area of the dotted line block. The training data set composes the sounding points and selected points from the isobathic data within the area. Tests are performed on RBF and LMBP network respectively. The results are shown in Fig. 5 and Fig. 6.

For the tests in realistic water area, there are no standard values for comparison besides the training data points. The verification is made directly on the training data points and the results are shown in Table 3.

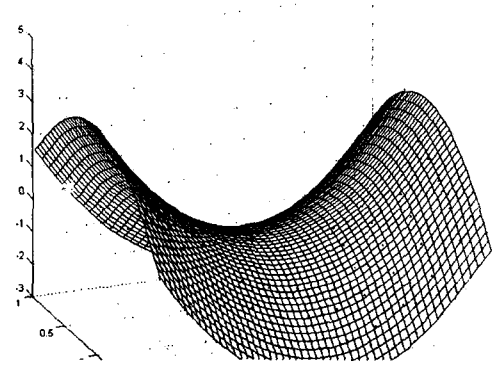

Fig. 3 Interpolation of hyperbolic paraboloid

The interpolated surface of RBF network strictly passes though the training data points. It illustrates a high-pass property. Even though the surface of the LMBP network usually does not pass through the training data points, it shows a good overall performance when the number of neurons on the hidden layer is properly chosen (Table 2). We can use either RBF or LMBP network depending on the specific practical purposes.

It should be noted that although the surface of RBF network pass through the training data points, it does not necessarily mean better overall performance. The high-pass property indicates poor behavior under disturbances. If there exists errors in the training data set, the interpolated value will be sensitively affected. Meanwhile, because the interpolation in its strict sense using RBF net work requires the number of neurons on hidden layer be identical to the number of training data points, tremendous memory space and computer time is needed when the training data set is large. Some of the modifications to the RBF network architecture can be taken into consideration.

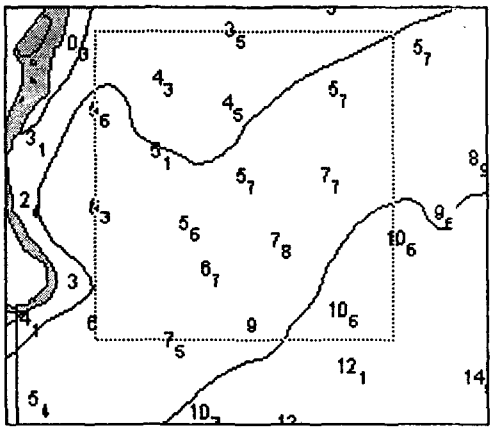

Fig. 4 Water area for interpolation

(1) When the charted data points are complicatedly distributed or the bottom surface is especially irregular, the eigenvalue of the interpolation matrix, Φ, may be very large and Φ is near to singular state. We can solve or improve such problem by introducing the regularization parameter, λ, and the free parameter vector is calculated by

$$\mathbf{w}=(\mathbf{G}+\lambda\mathbf{I})^{-1}\mathbf{d}$$

where $\mathbf{I}$ is a N-by-N identical matrix and $\mathbf{G}$ is called Green's matrix. The elements of $\mathbf{G}$ are determined by

$$G(\mathbf{x},\mathbf{x}_i)=\exp(-\frac{1}{2\sigma_i^2}\|\mathbf{x}-\mathbf{x}_i\|^2)$$

The value of λ is determined according to the practical applications. When λ=0, the strict interpolation is performed and by increasing λ, the smoothing effect of the interpolation is increased accordingly. In practice, we may always choose sufficiently large λ to ensure that $\mathbf{G}+\lambda\mathbf{I}$ is positive definite and therefore invertible.

(2) When the training data set is large, the free parameter vector can be determined by

$$\mathbf{w}=\mathbf{G}^{+}\mathbf{d}$$

where $\mathbf{G}^{+}=(\mathbf{G}^{T}\mathbf{G})^{-1}\mathbf{G}^{T}$, $\mathbf{G}$ is $(N\times m_0)$ green's matrix, N is the number of training data points and m_0 is the number

of neurons on the hidden layer of the RBF network.

Table 3 Test result of the water area

Chart depth	LMBP		RBF	
	Output	Error	Output	Error
5.70	5.64	*-0.06*	5.70	*0.0*
7.80	7.53	*-0.27*	7.80	*0.0*
7.70	7.25	*-0.45*	7.70	*0.0*
5.60	5.99	*0.39*	5.60	*0.0*
4.50	4.48	*0.02*	4.50	*0.0*
6.70	6.79	*0.09*	6.70	*0.0*
5.10	5.27	*0.17*	5.10	*0.0*
5.70	5.49	*-0.21*	5.70	*0.0*
9.00	8.67	*-0.33*	9.00	*0.0*
4.30	4.52	*0.22*	4.30	*0.0*
10.60	10.30	*-0.30*	10.60	*0.0*
10.60	9.95	*-0.65*	10.60	*0.0*
3.50	3.50	*0.00*	3.50	*0.0*
6.30	6.21	*-0.09*	6.30	*0.0*
7.50	7.44	*-0.06*	7.50	*0.0*
5.60	5.46	*-0.14*	5.60	*0.0*
5.00	4.93	*-0.07*	5.00	*0.0*
5.00	4.78	*-0.22*	5.00	*0.0*
5.00	5.20	*0.20*	5.00	*0.0*
10.00	9.70	*-0.30*	10.00	*0.0*
10.00	9.19	*-0.81*	10.00	*0.0*

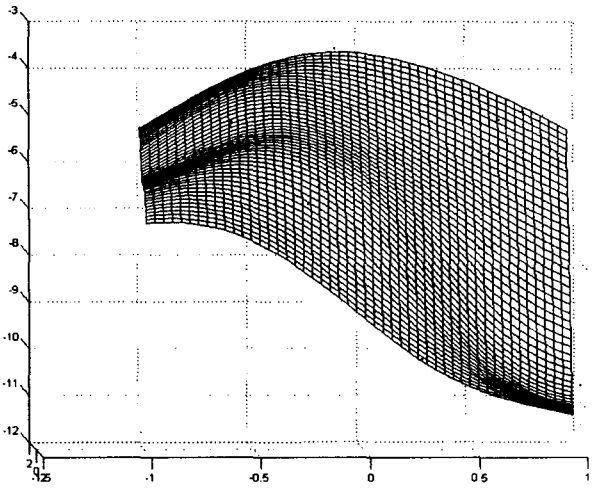

Fig.5 RBF interpolated surface

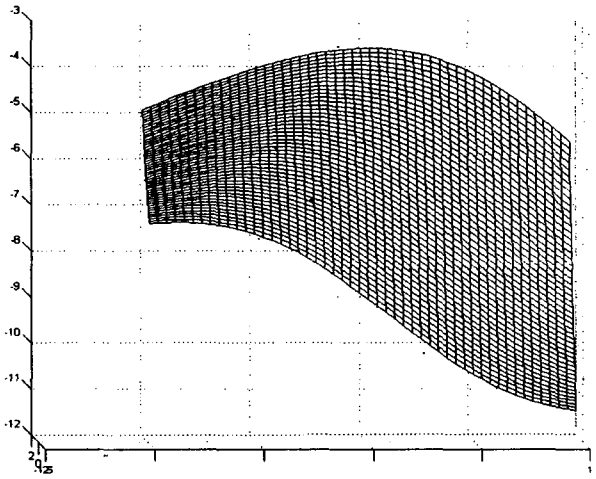

Fig.6 LMBP interpolated surface

3.3 Implementation Remarks

3.3.1 Partitioning of large areas

If the area processed by the network is too large or complicated, the effect of interpolation will decline. Therefore partition of the large areas is necessary. Either static or dynamic partitioning procedure can be implemented in accordance with the practical purposes. With static procedure, we divide a larger area into smaller sub-areas. Network training and interpolation are carried out for each of the sub-areas respectively. The results are synthesized to get the desired data, for instance, a depth data-grid. We can also arrange an array of networks for the whole area and train them properly before use. Then for arbitrary points in the area, the system search for right network for the interpolations. The static procedure works well for such applications as geographical modeling or information querying. In dynamic procedure, we build a network for an area around some moving point (e.g. present ship position). When the point moves near to the boundary of the area, subsequent area and network for interpolation will be built for further use. The dynamic procedure is suitable for monitoring ship movement as well as real-time simulation of shiphandling operations.

3.3.2 Normalization

Due to the characteristics of *logsig* transfer function and the radial-based functions, normalization of training

data set and input data is preferred. It will make a faster convergence in training and better behavior of the network. Usually we normalize the coordinate of the area block in domain of [-1.0, 1.0], and depth data, [0.0, 1.0].

3.3.3 Filtering isobathic data

In navigational char database, for example, IHO S-57 format database (IHO 1996), there are tow kinds of depth data, values of sounding points and values for depth contour (isobath). It is reasonable and necessary to include isobathic data in the interpolation. Along the contour the density of isobathic data is usually quite high, however, and the reliability and accuracy of these data are not necessarily higher than those of soundings. Therefore some algorithm of filtering the isobathic data is desired. One of our algorithms is to control the density of isobathic data to an similar level of that of sounding data. The isobathic data points is selected so that the following condition is satisfied,

$$\min(d \mid d \geq D/\sqrt{n})$$

where d=the distance between neighboring data points, D=diagonal distance of the interpolation area and n=the number of sounding points in the area.

4 Conclusions

Properly implemented, the neuron network approach is an effective way for charted depth interpolation. It is of significance either in information systems or real-time dynamic systems. The methods provided herewith plays good roles in several projects, such as navigation simulator development and assessment of harbor and waterway design, and successful and satisfied results have been achieved.

Although we focused on the interpolation of charted data. The methods discussed are also effective in similar applications. It helps solve the problems, for example, in calculating unevenly distributed elements of current or wind, wave field's distribution (Chen 2002), altitude in 3-D modeling (Hu 2003), land or sea surface temperature distribution, air pressure distribution etc.. Because of the versatile property of neuron networks, adaptation to different applications is effortless.

References

[1] Chen, D. and Wang, C., The Wave Field's Distribution Features of the Main Synoptic Systems Affecting China Sea, Journal of Shanghai Maritime University, Vol.23, No.1, 2002.3.

[2] Cressie, N., Statistics for spatial data: John Wiley & Sons, New York, 1993

[3] Hagan, M. T., Demuth, H. B. and Beale, M., Neural Network Design, PWS Publishing Company, U.S.A., 1996.

[4] Haykin, S., Neural Networks: A Comprehensive Foundation, 2nd Ed. Prentice-Hall, Inc., U. S. A., 1999

[5] Hu, S. and Shi, C., A Study on technical Design of Terrain with 3D in Shiphandling Simulators Development, Journal of Shanghai Maritime University, Vol.24, No.1, 2003.3.

[6] International Hydrographic Organization, IHO Transfer Standard For Digital Hydrographic Data, Publication S-57, Edition 3.0, March 1996

[7] Micchelli,C.A., Interpolation of Scattered Data: Distance Matrices and Conditionally Positive Definite Functions. Constructive Approximation, Vol.2, 1986.

[8] Shi, C., Application and Functional Requirements of Simulator in Harbor and Waterway Design, Journal of Korean Navigation and Port Research, Vol.26, No.1, March 2002

[9] Wu, X., Principles and Methods of Geographic Information Systems, Publishing House of Electronic Industry, Beijing, 2002

[10] Xiao, Y., A Study of Ship Form Through Simulation Tests for Shanghai Deep Port, Journal of Shanghai Maritime University, Vol.23, No.3, 2002.9.

[11] Zimmerman, D., An Experimental Comparison of Ordinary and Universal Kriging and Inverse Distance Weighting, Mathematical Geology, Vol. 31, No. 4, 1999