Neural Networks, Vol. 5, pp. 441-457, 1992
Printed in the USA. All rights reserved.

0893-6080/92 $5.00 + .00

ORIGINAL CONTRIBUTION

Modified Hebbian Learning for Curve and Surface Fitting

LEI XU[1,2]*, ERKKI OJA[3], AND CHING Y. SUEN[2]

[1]Harvard Robotics Laboratory, Harvard University [2]Centre for Pattern Recognition & Machine Intelligence, Concordia University
[3]Lappeenranta University of Technology, Department of Information Technology

(*Received* 15 *March* 1991; *revised and accepted* 11 *November* 1991)

Abstract—*A linear neural unit with a modified anti-Hebbian learning rule is shown to be able to optimally fit curves, surfaces, and hypersurfaces by adaptively extracting the minor component (i.e., the counterpart of principal component) of the input data set. The learning rule is analyzed mathematically. The results of computer simulations are given to illustrate that this neural fitting method considerably outperform the commonly used least square fitting method in resisting both normal noise and outlier.*

Keywords—Hebbian learning, Principal component analysis, Minor components, Curve and surface fitting, Total least square method, Stochastic approximation.

1. INTRODUCTION

For building a neural network, linear units are the simplest ones. In spite of its apparent simplicity, a linear unit is able to do some things which are quite nontrivial. It has been found (Oja, 1982) that a simple linear unit with an unsupervised constrained Hebbian learning rule can extract the principal component from stationary input data. Later, Linsker (1986a, 1986b, 1986c) showed that in a layered feedforward network of linear units with random inputs and Hebbian learning rule, spatial opponent units, orientation selective units, and orientation columns emerge in successive layers in a way which resembles what is observed in the mammalian primary visual cortex.

Recently, there has been increasing interest in the study of unsupervised learning networks consisting of linear units. Several new developments have been made. First, some neural-net models have been proposed to extract $k > 1$ principal components or the principal component subspaces (Foldiak, 1989; Oja, 1989; Rubner & Tavan, 1989; Rubner & Schulten, 1990; Sanger, 1989) instead of the first principal component described in an earlier work (Oja, 1982). Second, the learning problem of optimizing the usual quadratic error function E of linear feedforward neural networks in autoassociation mode have been related to principal component analysis and it has been shown that E of such a network has no local minima (Baldi & Hornik, 1989; Chauvin, 1989). Third, neural networks for principal components analysis have been extended to neural networks for constrained principal analysis which can avoid the undesirable redundant components and noisy components (Kung, 1990; Kung & Diamantaras, 1990). A good recent review on many of these developments is given by (Baldi & Hornik, 1991). All these results demonstrate the deep connection between linear neural nets, Hebbian learning, and the well known principal component analysis (PCA) of statistics.

The PCA approach originally appeared in multivariate statistics literature. Closely related to the Karhunen-Loeve (K-L) transform, it has been widely studied and used in signal processing and pattern recognition for data compression and feature extraction. In PCA, the principal components (i.e., directions in which the data have the largest variances) are regarded as important while those components which have small variances, for convenience called in the following the *minor components,* are often regarded as unimportant or noise. However, in some cases the minor components are of the same importance as principal components, e.g., they have been used in signal processing in the Pisarenko method of spectral estimation (Karhunen, 1982).

In this paper, we will show that the minor components can play an important role in a classical statistical problem: curve or hypersurface fitting. The problem is often encountered in many engineering applications as

---

* On leave from Department of Mathematics, Peking University, Beijing, P. R. China From February 1989–May 1990, he was with Lappeenranta University of Technology, Finland.

The work was supported partly by Tekes Grant 4196 under the Finsoft Project of Finland, and partly by the Natural Sciences and Engineering Research Council of Canada, and the National Networks of Centres of Excellence program of Canada.

well as cognitive perception problems (e.g., computer vision). The conventional method for such tasks is the usual Least Square (LS) fitting method. However, in many cases, LS is suboptimal, and the optimal least square method is the so called Total Least Square (TLS) method. This paper will show that the problems of optimal fitting in TLS sense can be described as *Minor Components Analysis* (MCA) problems and a linear neural unit modified from the unit given in Oja (1982) can solve the problems. Computer simulations show that this neural fitting method outperform LS significantly.

2. OPTIMAL FITTING PROBLEMS

The problems of using a model of line (curve), plane (surface), and hyperplane (hypersurface) to fit a given data set are often encountered in many engineering applications as well as in computer vision. Usually, the LS method is used to solve such problems. For example, given a set of data points $D_x = (\{x_1^{(i)}, x_2^{(i)}), i = 1, 2, \ldots, N\}$, the problem of having a line model $x_2 = kx_1 + d$ to fit D_x in the usual LS sense becomes the problem of finding a pair of estimates $\hat{k}, \hat{d}$ such that

$$E_2(\hat{k}, \hat{d}) = \min_{k,d} \{E_2(k, d)\},$$

$$E_2(k, d) = \sum_{i=1}^{N} \epsilon_i^2, \quad \epsilon_i = (x_2^{(i)} - kx_1^{(i)} - d). \tag{1}$$

Referring to Figure 1, there $|\epsilon_i|$ is the absolute length of a vertical bar from point $p_i = (x_1^{(i)}, x_2^{(i)})$ to the fitted line, and thus eqn (1) means "to minimize the sum of the squared lengths of all the vertical bars." In fact, this implicitly contains an assumption that only the measurements $x_2^{(i)}$ contain errors while the measurements $x_1^{(i)}$ are accurate. However, in many cases such as in image recognition and computer vision, all the measurements contain a certain degree of errors. In such cases, a line $x_2 = \hat{k}x_1 + \hat{d}$ obtained in the usual LS sense by eqn (1) is not optimal. The optimal way should be to minimize "the sum of the squared lengths of all the bars which are perpendicular to the estimated line" (see Figure 1) (i.e., to minimize the sum of all the squared distances r_i's from point p_i to the estimated line).

$$E_2(\hat{k}, \hat{d}) = \min_{k,d} \{E_2(k, d)\},$$

$$E_2(k, d) = \sum_{i=1}^{N} r_i^2, \quad r_i = \frac{|x_2^{(i)} - kx_1^{(i)} - d|}{\sqrt{1 + k^2}}. \tag{2}$$

This is the basic idea of the so called TLS method (see e.g., Golub & Van Loan, 1983). In comparison with the usual LS method, to obtain the solution of TLS is generally quite burdensome. Equation (1) can be reduced to a problem of solving linear equations, while eqn (2) results in a problem of solving a third order nonlinear equation of k. For the more general

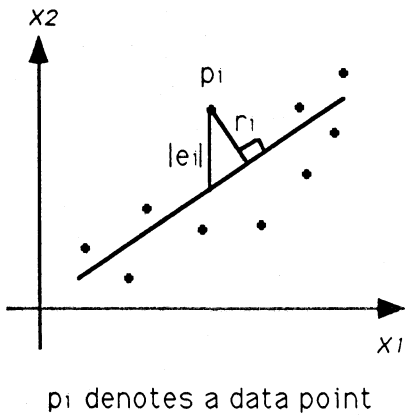


FIGURE 1. To fit a set of data points by a line, LS minimizes the sum of the squared lengths of every vertical bar $|e_i|$, while TLS minimizes the sum of the squared length of every bar r_i which is perpendicular to the estimated line.

problems which involve a large number of variables, the task becomes more complicated. This is probably why TLS has not been as widely used as the usual LS method although the basic idea of TLS was proposed long ago (Pearson, 1901).

However, in the case of line or hyperplane fitting, when the line or hyperplane models are expressed as

$$a_1x_1 + a_2x_2 + c_0 = 0, \tag{3a}$$

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n + c_0 = 0, \tag{3b}$$

where $x_i, i = 1, \ldots, n$ are variables and c_0 is an arbitrary constant, we will show that the problem of optimal fitting in the TLS sense is not so intricate and its solution can be obtained by a neural unit via adaptive learning. First, again let us take the problem of line fitting as an example. For eqn (3a), the TLS fitting problem is to minimize the following total least square error E

$$E = \sum_{i=1}^{N} r_i^2, \quad r_i = \frac{|a_1x_1^{(i)} + a_2x_2^{(i)} + c_0|}{\sqrt{a_1^2 + a_2^2}}. \tag{4a}$$

Let $\mathbf{a} = [a_1, a_2]^T$ and $\mathbf{x_i} = [x_1^{(i)}, x_2^{(i)}]$. Then E can be further expressed as

$$E = \sum_{i=1}^{N} \frac{(\mathbf{a}^T\mathbf{x_i} + c_0)^2}{\mathbf{a}^T\mathbf{a}} = N\frac{\mathbf{a}^TR\mathbf{a} + 2c_0\mathbf{a}^T\mathbf{e} + c_0^2}{\mathbf{a}^T\mathbf{a}},$$

$$R = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i}\mathbf{x_i}^T, \quad \mathbf{e} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i}, \tag{4b}$$

where $\mathbf{e}, R$ are the mean vector and the autocorrelation matrix of data set D_x. From $dE/d\mathbf{a} = 0$, the critical points of E should satisfy

$$R\mathbf{a} + c_0\mathbf{e} - \lambda\mathbf{a} = 0, \quad \lambda = \frac{\mathbf{a}^TR\mathbf{a} + 2c_0\mathbf{a}^T\mathbf{e} + c_0^2}{\mathbf{a}^T\mathbf{a}}. \tag{4c}$$

In general, eqn (4c) is difficult to solve because it is a third order matrix equation.

Here, we use a special strategy for solving the equa-

tion. First, by taking expectation on both sides of eqn (3b), we can obtain

$$c_0 = -\mathbf{a}^T\mathbf{e}, \tag{5a}$$

which we substitute into eqn (4c) and simplify the equation into

$$\Sigma\mathbf{a} - \lambda\mathbf{a} = 0, \quad \lambda = \frac{\mathbf{a}^T\Sigma\mathbf{a}}{\mathbf{a}^T\mathbf{a}}, \tag{5b}$$

where $\Sigma = R - \mathbf{e}\mathbf{e}^T$ is the covariance matrix of data set D_x. So, we see that the TLS problem of eqn (4a) can be reduced to the problem of finding the minimum eigenvalue and its corresponding normalized eigenvector of matrix Σ, or in other words, finding the first minor component of data set D_x.

It is not difficult to see that for plane and hyperplane expressed by eqn (3b), if we let $\mathbf{a} = [a_1, a_2, \ldots, a_n]^T$ and $\mathbf{x_i} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}]$, eqns (4b and c) and eqns (5a and b) will also hold.

Generally speaking, the same technique applies to curves and hypersurfaces expressed as

$$a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \ldots + a_m f_m(\mathbf{x}) + c_0 = 0, \quad \mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \tag{6a}$$

where $f_i(\mathbf{x})$ is a function of $\mathbf{x}$ (e.g., quadratic curves)

$$a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + a_4 x_1 + a_5 x_2 + c_0 = 0. \tag{6b}$$

If we first transform each $\mathbf{x_i}$ into $\mathbf{f_i} = [f_1(\mathbf{x_i}), f_2(\mathbf{x_i}), \ldots, f_m(\mathbf{x_i})]^T$, we can obtain the same equations as eqns (4b and c) and eqns (5a and b) for the problems of TLS hypersurface fitting.

3. NEURAL UNIT AS AN OPTIMAL FITTING ANALYZER

Consider the linear neural unit shown in Figure 2 with inputs $\mathbf{x(t)} = [\xi_1(t), \ldots, \xi_n(t)]^T$, weights $\mathbf{m}(t) = [\mu_1(t), \ldots, \mu_n(t)]^T$ and output

$$z(t) = \sum_{i=1}^{n} \mu_i(t)\xi_i(t) = \mathbf{m}^T(t)\mathbf{x}(t), \tag{7a}$$

it has been shown (Oja, 1982) that by using a constrained Hebbian learning rule as follows

$$\mu_i(t+1) = \mu_i(t) + \alpha(t)(z(t)\xi_i(t) - z^2(t)\mu_i(t)) \quad \text{or}$$

$$\mathbf{m}(t+1) = \mathbf{m}(t) + \alpha(t)[z(t)\mathbf{x}(t) - z^2(t)\mathbf{m}(t)], \tag{7b}$$

the unit learns to function as a principal component analyzer of the stationary input vector stream $\mathbf{x}(t)$, in the sense that the weight vector $\mathbf{m}(t)$ tends asymptotically to the principal eigenvector of the input data correlation matrix. In (7b), $\alpha(t)$ is a positive scalar gain parameter that must be chosen in a suitable way.

Here, we still use the same linear unit, but change eqn (7b) into a constrained anti-Hebbian learning rule given either by

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t)z(t)[\mathbf{x}(t) - z(t)\mathbf{m}(t)] \tag{8a}$$

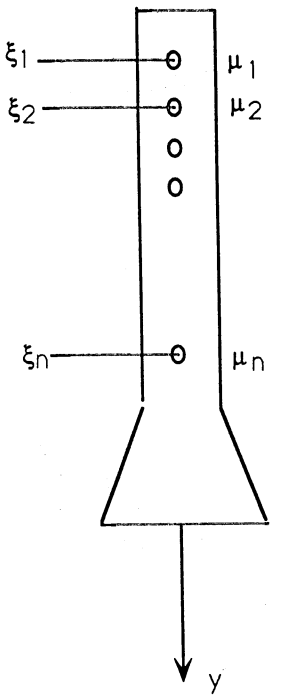


FIGURE 2. A linear neural unit.

or its explicitly normalized version

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t)z(t)\left[\mathbf{x}(t) - \frac{z(t)\mathbf{m}(t)}{\mathbf{m}^T(t)\mathbf{m}(t)}\right]. \tag{8b}$$

Then substituting eqn (7a) into the above equations produces, respectively,

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t) \times [\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t) - \mathbf{m}^T(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t)\mathbf{m}(t)] \tag{8c}$$

and

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t) \times \left[\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t) - \frac{\mathbf{m}^T(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t)}{\mathbf{m}(t)^T\mathbf{m}(t)}\mathbf{m}(t)\right]. \tag{8d}$$

The techniques of stochastic approximation theory are available for analyzing the learning rules of the type given here (see Kushner & Clark 1978; Ljung 1977). Without going rigorously into details, it can be shown that if the distribution of $\mathbf{x}(t)$ satisfies some not unrealistic assumptions and the learning gain $\alpha(t)$ is not constant but allowed to decrease to zero in an appropriate way (e.g., proportionally to $1/t$), then eqn (8c) and eqn (8d) can be approximated by the following differential equations:

$$\frac{d\mathbf{m}(t)}{dt} = -[\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t) - \mathbf{m}^T(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t)\mathbf{m}(t)],$$

and

$$\frac{d\mathbf{m}(t)}{dt} = -\left[\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t) - \frac{\mathbf{m}^T(t)\mathbf{x}(t)\mathbf{x}^T(t)\mathbf{m}(t)}{\mathbf{m}(t)^T\mathbf{m}(t)}\mathbf{m}(t)\right].$$

Furthermore, assume that the input vector $\mathbf{x}(t)$ stays stationary throughout the learning period, and $\mathbf{x}(t)$, $\mathbf{m}(t)$ are statistically independent, by taking averages in the both sides of these two equations, we have the corresponding averaging differential equations

$$\frac{d\mathbf{m}(t)}{dt} = -R\mathbf{m}(t) + [\mathbf{m}(t)^T R\mathbf{m}(t)]\mathbf{m}(t) \quad (8e)$$

$$\frac{d\mathbf{m}(t)}{dt} = -R\mathbf{m}(t) + \frac{\mathbf{m}(t)^T R\mathbf{m}(t)}{\mathbf{m}(t)^T\mathbf{m}(t)}\mathbf{m}(t), \quad (8f)$$

where $R = E(\mathbf{x}\mathbf{x}^T)$ is the autocorrelation matrix of inputs. The approximations of eqn (8e) to eqn (8c) and eqn (8f) to eqn (8d) are in this sense that the asymptotic paths of eqn (8e) and eqn (8c) (or eqn (8f) and eqn (8d), respectively) are close with a large probability and eventually the solutions $\mathbf{m}(t)$ of eqn (8c) and eqn (8d) tend with probability one to the uniformly asymptotically stable solutions of eqn (8e) and eqn (8f), respectively (see Oja, 1982; Oja & Karhunen, 1985 for details).

To study the asymptotic stability of eqns (8e, f) we have from $d\mathbf{m}(t)/dt = 0$ that

$$R\mathbf{m}(t) = [\mathbf{m}(t)^T R\mathbf{m}(t)]\mathbf{m}(t) \quad \text{and}$$

$$R\mathbf{m}(t) = \frac{\mathbf{m}(t)^T R\mathbf{m}(t)}{\mathbf{m}(t)^T\mathbf{m}(t)}\mathbf{m}(t), \quad (8g)$$

that is, all the eigenvectors of R are the fixed points of eqns (8e, f). By expanding $\mathbf{m}(t)$ in terms of eigenvectors in the way similar to (Oja, 1982; Oja & Karhunen, 1985), the following theorem can be proven.

THEOREM 1. *Let R be positive semidefinite with the minimum eigenvalue of multiplicity one, and let $\lambda_{\min}$ and $\mathbf{c}_{\min}$ be the minimum eigenvalue and its correspondent normalized eigenvector of R. If $\mathbf{m}(0)^T\mathbf{c}_{\min} \neq 0$, then*

1. *For eqn (8e), asymptotically $\mathbf{m}(t) = \eta(t)\mathbf{c}_{\min}$ with $\eta(t)$ a scalar function, in other words, $\mathbf{m}(t)$ has the same direction as $\mathbf{c}_{\min}$. In the special case of $\lambda_{\min} = 0$, it holds $\lim_{t\to\infty}\mathbf{m}(t) = (\mathbf{m}(0)^T\mathbf{c}_{\min})\mathbf{c}_{\min}$.*
2. *For eqn (8f), it holds that*

$$\lim_{t\to\infty} \mathbf{m}(t) = \mathbf{c}_{\min} (\text{or } -\mathbf{c}_{\min})$$

$$\lim_{t\to\infty} \mathbf{m}(t)^T R\mathbf{m}(t) = \lambda_{\min} = \min_{\mathbf{m}} \{\mathbf{m}^T R\mathbf{m}\}. \quad (9)$$

See the Appendix for the proof of the theorem.

From the results of Section 2 (e.g., eqns (4a) and (5b)), one will see that the neural unit of Figure 2 with learning equation either eqn (8a) or (8b) can be directly used as an optimal TLS fitting analyzer if $\mathbf{e} = E(\mathbf{x}) = 0$ for input data set D_x. In this case, eqn (5a) shows that $\mathbf{e} = 0$ results in $c_0 = 0$, and it follows from eqn (3b) and eqn (5b) that any vector which has the same direction as $\mathbf{c}_{\min}$ is a solution which minimizes the total least square error of eqn (4a). Thus, we can take the unit vector of the direction as the final solution (i.e., let $\mathbf{a}(t) = \mathbf{m}(t)/\|\mathbf{m}(t)\|$). From the above theorem, $\mathbf{a}(t)$ approaches the TLS solution as t becomes large enough.[1] Specifically, if eqn (8b) is used, after the stabilization of the learning, when a new data point $\mathbf{x}$ is input to the neural unit of Figure 2, the unit can also automatically detect not only the distance between the point and the fitted hyperplane simply through the magnitude of the output $|y|$, but also which side of the hypersurface the point is on through the sign of the output $\text{sign}[y]$.

In the case that $\mathbf{e} = E(\mathbf{x}) \neq 0$, the above unit can not be used directly. However, noticing that $\Sigma = R - \mathbf{e}\mathbf{e}^T = E[(\mathbf{x} - \mathbf{e})(\mathbf{x} - \mathbf{e})^T]$, we see that a slight preprocessing of subtracting $\mathbf{e}$ from each data point can make all the above discussions remain true. The only extra issue here is that the representation of the fitted hyperplane needs not only the obtained final solution $\mathbf{a}$ alone but also an accompanied parameter $c_0 = -\mathbf{m}^T\mathbf{e} \neq 0$.

4. COMPUTER SIMULATIONS ON LINE FITTING

We generate a data set $D_x = \{(x_i, y_i), i = 1, \ldots, 500\}$ as shown in Figure 3(a). The problem is to use a parameterized line model (e.g., $a_1x + a_2y = 1$) to fit the data set so that the right parameters a_1, a_2 are solved. On this pure data set, both the usual LS method and our proposed neural method can give the perfect solution, so this simple case is not of our interest here. Instead, we consider the case when Gaussian noise is added to D_x given by

$$x_i' = x_i + n_x, \quad y_i' = y_i + n_y, \quad i = 1, \ldots, 500, \quad (10a)$$

where $E[n_x, n_y]^T = [0, 0]^T$, $Cov[n_x, n_y] = \sigma^2 I_{2\times 2}$ and $I_{2\times 2}$ is a 2×2 unit matrix. Figure 3(b) shows the noise-disturbed data set $D_x' = \{(x_i', y_i'), i = 1, \ldots, 500\}$ with noise variance $\sigma^2 = 0.5$. Computer simulations are conducted on the noisy set, using both the LS method and the above neural method.

For the LS method, we try both of the two possible versions of parameterization:

1. Version LS1: we first use the LS method with parameterization $y = kx + d$ (i.e., we get the solution $\mathbf{s} = [k, d]^T$ of the normal equations as)

$$\mathbf{s} = (X^TX)^{-1}X^T\mathbf{y}', \quad \mathbf{y}' = [y_1', \ldots, y_{500}']^T$$

$$X^T = \begin{pmatrix} x_1' & \cdots & x_{500}' \\ 1 & \cdots & 1 \end{pmatrix},$$

and then transform $\mathbf{s}$ into

$$a_1 = -k/d, \quad a_2 = 1/d.$$

[1] When eqn (8a) is used, although it is shown in the Appendix by eqn (A4) that the magnitude of $\mathbf{m}(t)$ tends either to zero or infinity when $\lambda_{\min} > 0$, here the magnitude of $\mathbf{a}(t)$ stays at 1 because it is a unit vector along the direction of $\mathbf{m}(t)$. When eqn (8b) is used, according to eqn (9) we can also directly let $\mathbf{a}(t) = \mathbf{m}(t)$ since in this case the magnitude of $\mathbf{m}(t)$ tends to the magnitude of $\mathbf{c}_{\min}$.

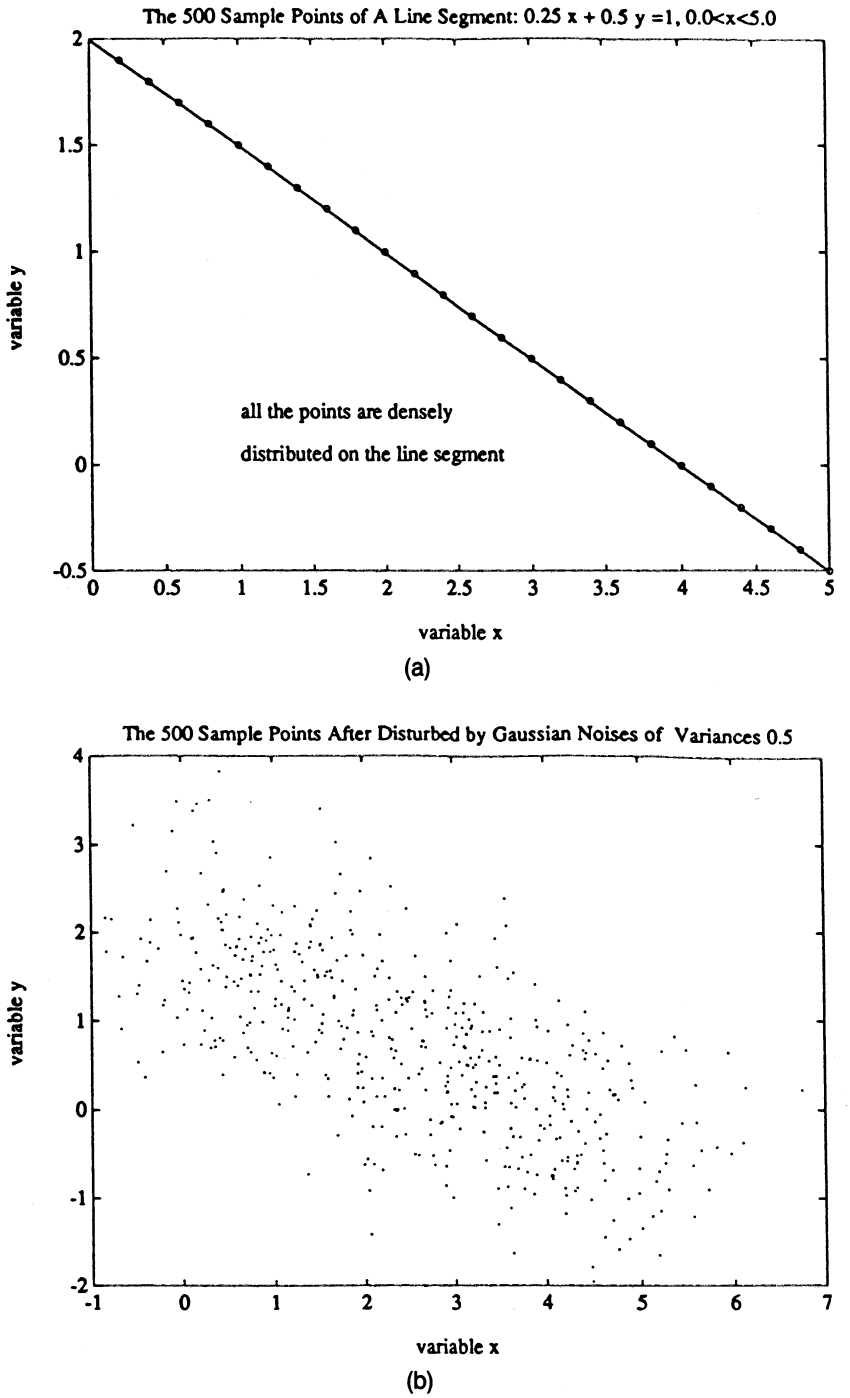


FIGURE 3. The data set used in line fitting simulations. (a) The 500 pure sample points come from the line segment by uniform sampling; for clarity, only a portion of sample points are shown here. (b) The 500 noise-disturbed sample points obtained by using Gaussian noise with variance $\sigma^2 = 0.5$ in the way given by eqn [10a].

2. Version LS2: we obtain the solution $\mathbf{a} = [a_1, a_2]^T$ directly from parameterization $a_1 x + a_2 y = 1$ by

$$\mathbf{a} = (X^T X)^{-1} X^T \mathbf{b}, \quad \mathbf{b} = [1, \ldots, 1]^T$$

$$X^T = \begin{pmatrix} x'_1 & \cdots & x'_{500} \\ y'_1 & \cdots & y'_{500} \end{pmatrix}.$$

For our Neural Total Least Square (NTLS) fitting method, the simulations have been implemented in the following steps:

1. Compute means $\mathbf{e} = [e_1, e_2]^T$, $e_1 = \frac{1}{500} \sum_{i=1}^{500} x'_i$, $e_2 = \frac{1}{500} \sum_{i=1}^{500} y'_i$.
2. Subtract the means from each data point, i.e., $\xi_1^{(i)} = x'_i - e_1$, $\xi_2^{(i)} = y'_i - e_2$.
3. Initialize weight vector $\mathbf{m} = [\mu_1, \mu_2]^T$ by a random number from a uniform distribution on [0, 1] $\times$

[0, 1], and let $\mathbf{x} = [\xi_1^{(j)}, \xi_2^{(j)}]^T$, j be a random integer with equal probability being any integer of $\{1, 2, \ldots, 500\}$.

4. Use learning rule (8a) to adaptively adjust $\mathbf{m}$, i.e.,

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t)z(t)[\mathbf{x} - z(t)\mathbf{m}(t)], \quad (10b)$$

where $z(t) = \mathbf{m}(t)^T\mathbf{x}$ and $\alpha(t)$ is the learning rate.

5. Calculate the current solution $\mathbf{a}(t+1) = [a_1(t+1), a_2(t+1)]^T$ by transforming equation $[x, y]\mathbf{m} + c_0 = 0$ into $a_1(t+1)x + a_2(t+1)y = 1$, i.e., let

$$\mathbf{a}(t+1) = -\mathbf{m}(t+1)/c_0, \quad c_0 = \mathbf{m}(t+1)^T\mathbf{e}. \quad (10c)$$

Remarks. (a) In step (4), theoretically, the best selection of $\alpha(t)$ should be that given by the Robbins-Monro stochastic approximation procedure (Oja & Karhunen, 1985; Robbins & Monro, 1951). For simplicity, here we just let α start at 0.01 and linearly reduce it to 0.0025 at the first 500 learning steps (i.e., t from 1 to 500), and hereafter keep it unchanged at 0.0025. The results given here show that this simple selection already gives acceptable results although some further improvement could be expected by using a more sophisticated selection. Such improvements are introduced in the next section. (b) One can also use learning rule (8b) in step (4), i.e., to replace eqn (10b) by eqn (10d)

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha(t)z(t)\left[\mathbf{x} - \frac{z(t)\mathbf{m}(t)}{\mathbf{m}^T(t)\mathbf{m}(t)}\right]. \quad (10d)$$

(c) If the parameterization $a_1x + a_2y = 1$ is used, then in step (5) $\mathbf{a}(t)$ given by eqn (10c) is not a unit vector $\mathbf{a}(t) = \mathbf{m}(t)/\|\mathbf{m}(t)\|$ as discussed in Section 4. Here, it follows from Theorem 1 that asymptotically $\mathbf{a}(t) = \eta(t)\mathbf{c}_{\min}/\eta(t)\mathbf{e}^T\mathbf{c}_{\min} = \kappa\mathbf{c}_{\min}$ where $\kappa = 1/\mathbf{e}^T\mathbf{c}_{\min}$ is a constant. In other word, asymptotically, $\mathbf{a}(t)$ is also a stable TLS fitting solution.

The experimental results are given in Table 1 and Figures 4(a–c). It can be seen from Table 1 and Figure 4(a) that NTLS gives the best result and LS1 gives the worst. The fast learning process of NTLS is illustrated in Figure 4(b). The learning process of the same problem but with different randomly chosen initial parameter estimates is shown in Figure 4(c). Its average solution over the last 500 steps is $a_1 = 0.2588$, $a_2 = 0.4834$, with variances 0.36×10^{-6} and 0.324×10^{-5}. The solution at the last step is $a_1 = 0.2588$, $a_2 = 0.4833$. They are almost the same as those given in Table 1. From many trials with different initial estimates, we have found that the initialization only has an influence on the learning speed, but it has nearly no influence on the converged results, as also indicated by Theorem 1.

TABLE 1
The Solutions Obtained by Different Methods On The Noisy Data Set. For NTLS, the Solution is Given in Two Ways: (a) the Estimates of a_1, a_2 at the 3,000-th (i.e., the last) Learning Step, (b) the Average Value of a_1, a_2 at the Steps Between 2,500 and 3,000 (see Figure 4b) Together with the Correspondent Varianes

	a_1	a_2
Original	0.25	0.5
LS1	0.2536	0.3465
LS2	0.2356	0.5607
NTLS (last)	0.2589	0.4832
NTLS (means)	0.2589	0.4833
NTLS (vars.)	0.36×10^{-6}	0.324×10^{-5}

It may be useful to indicate that here we have not discussed how to terminate the learning process. There are several ways to do it, e.g., to stop after a prespecified number of steps or when the fluctuations fall within an error bound.

5. IMPROVEMENT ON THE LEARNING RATES OF NTLS LEARNING

Theorem 1 in the previous section only deals with the asymptotic solutions of the learning rules (10b and 10d). It says nothing about finite-time behavior, especially, about the speed of learning, which is strongly affected by the specific choice of the gain α. The theoretical correspondence between the discrete-time learning algorithms and the continuous averaged differential equations requires that the gain sequence $\alpha = \alpha(t)$ goes to zero. In practice, however, a large α gives faster convergence. It turns out that in practice the learning rule of eqn (10b), using constant or prespecified gain α, may not behave well even after the instability of the norm of $\mathbf{m}(t)$ is removed by using $\mathbf{a}(t) = \mathbf{m}(t)/\|\mathbf{m}(t)\|$ for the parameter vector. The learning rate α should be quite small, otherwise the learning will become unstable and diverge. This may bring some problems (e.g., (a) a small learning rate gives a low learning speed, (b) one should pay efforts on selecting a suitable learning rate in order to prevent learning divergence, (c) the degree of fluctuation and thus the solution's accuracy will also be affected by an inappropriately predefined learning rate).

The reason for this problem is given by the following proposition (see the Appendix for the proof):

PROPOSITION 1. *In eqn (10b), let*

$$r' = |\mathbf{m}(t+1)^T\mathbf{x}|^2/\|\mathbf{m}(t+1)\|^2 \quad \textit{and}$$
$$r = |\mathbf{m}(t)^T\mathbf{x}|^2/\|\mathbf{m}(t).\|^2$$

It holds that $r' > r$ *if* $\alpha > \alpha_b = 2/\|\mathbf{x}\|^2(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{xm})$ *and* $|\cos\theta_{\mathbf{xm}}| < 1/\sqrt{2}\|\mathbf{m}(t)\|$, *where* $\theta_{\mathbf{xm}}$ *is the angle between the directions of* $\mathbf{x}$ *and* $\mathbf{m}(t)$.

During the learning process of using eqn (10b) on a noisy data set, it is possible for some $\mathbf{x}$ to have a large magnitude and a large angle with respect to the direction of $\mathbf{m}(t)$. In such a case $\alpha_b < 1$ is a small positive number and $|\cos\theta_{\mathbf{xm}}| < 1/\sqrt{2}\|\mathbf{m}(t)\|$; it is then possible for α (which varies within the real interval (0, 1)) to be larger than α_b. Since r' and r in fact give the distances from x perpendicular to the present fitted hy-

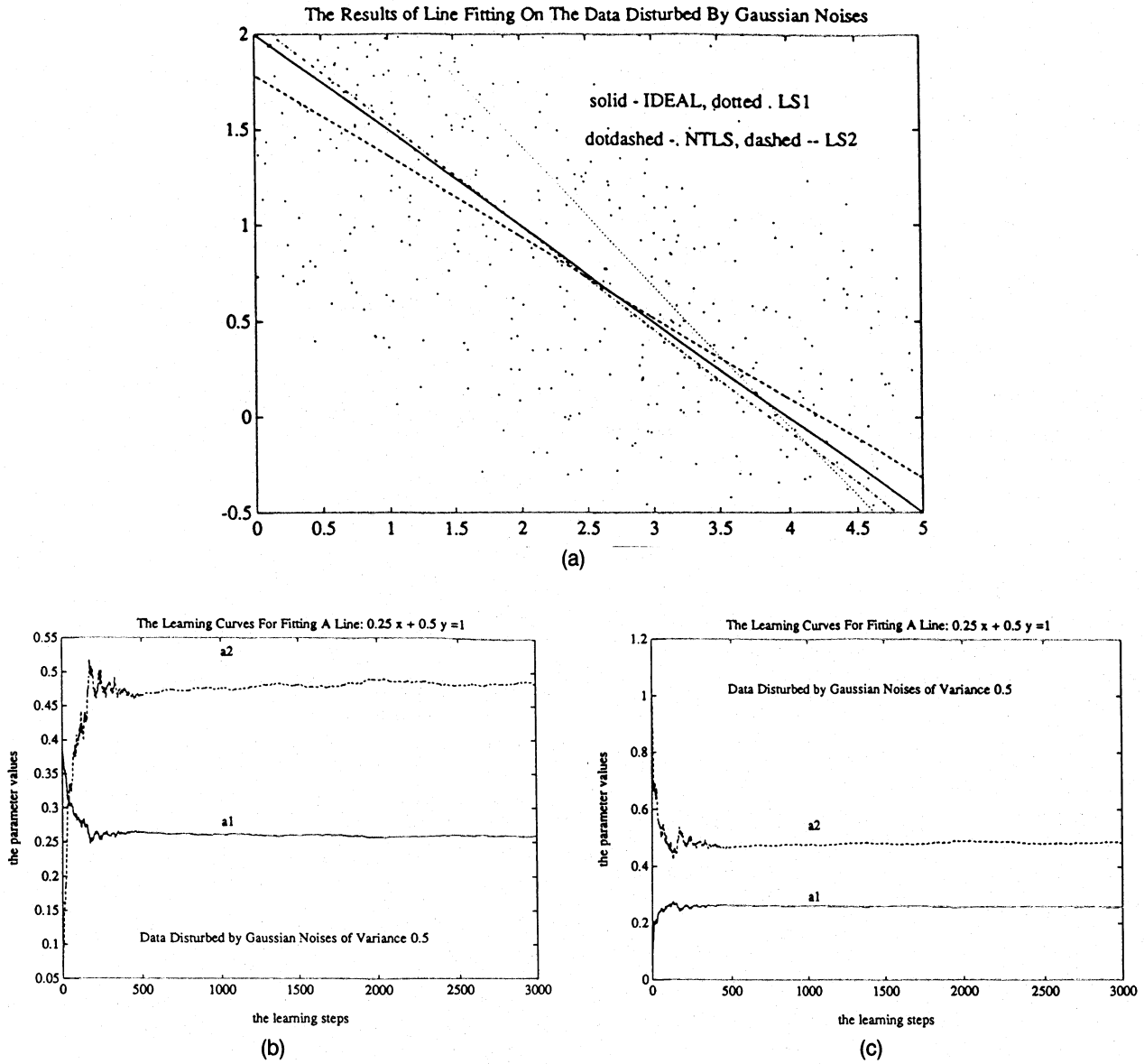


FIGURE 4. The results of line fitting on the data set given in Figure 3(b). (a) The lines of the fitting results by LS1, LS2, and NTLS are visualized in comparison with the ideal (i.e., original) line which is plotted as the solid line. (b) The learning process of NTLS: during the first 500 learning steps, the estimates are rapidly modified from a random initial value to the right solution, and hereafter they have only slight fluctuations. (c) Another learning process of the same problem but with different randomly chosen initial estimates. It results in the converged values almost the same as those in (b).

perplane after and before a modification, it follows that eqn (10b) may sometimes increase the estimation error due to disturbance caused by noisy data. When this disturbance is too large, it will make $\mathbf{m}(t)$ deviate drastically from the normal learning, which may result in divergence or a increased learning time.

Similarly, for the learning eqn (10d) we have the following proposition (see also the Appendix for the proof):

PROPOSITION 2. *In eqn (10d), let r', r be the same as in Proposition* 1. *It holds that* $r' > r$ *if* $\alpha > \alpha_b(\mathbf{x}, \mathbf{m}(t))$, *where* $\alpha_b(\mathbf{x}, \mathbf{m}(t))$ *is a real number, depending on* $\mathbf{x}$, $\mathbf{m}(t)$, *which can take negative values or small positive values smaller than* 1.

This proposition indicates that for a noisy data set the use of eqn (10d) may have problems similar to what happens on the learning process eqn (10b), although asymptotically the solution will be stable if α goes slowly to zero.

To amend the above discussed problems, it is possible to choose the gain sequence in a data-dependent way to guarantee faster learning. Equation (11a) is given as an improved version of eqn (10b) and eqn (11b) as an improved version of eqn (10d):

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha\alpha_a z(t)[\mathbf{x} - z(t)\mathbf{m}(t)]$$
$$z(t) = \mathbf{m}(t)^T\mathbf{x}, \quad \alpha_a = 1/|\|\mathbf{x}\|^2 - z^2(t)|, \quad (11a)$$

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha\alpha_a z(t)\left[\mathbf{x} - \frac{z(t)\mathbf{m}(t)}{\mathbf{m}^T(t)\mathbf{m}(t)}\right]$$
$$z(t) = \mathbf{m}(t)^T\mathbf{x},$$
$$\alpha_a = 1/\left(\|\mathbf{x}\|^2 - \frac{z^2(t)}{\|\mathbf{m}(t)\|^2}\right) = \frac{1}{\|\mathbf{x}\|^2\sin^2\theta_{\mathbf{xm}}}. \quad (11b)$$

Now the learning rate consists of two factors: one is α which can still be chosen in a data-independent way like in eqn (10b) and eqn (10d), and the other is $\alpha_a > 0$ which is adaptively adjusted to suit the present state and input.

For eqn (11a) and eqn (11b), we can prove the theorem given below (see the Appendix for the proof):

THEOREM 2. *In eqn (11a) or (11b), let $\rho(\alpha) = r'/r$, with r', r the same as in Proposition 1. For any input sample $\mathbf{x}(t)$, $\rho(\alpha)$ monotonically decreases as α monotonically increases from 0 to 1. Specifically, we also have that $\rho(\alpha)$ monotonically decreases from 1 to 0 as α monotonically increases from 0 to 1 for either the case when eqn (11b) is used or the case when eqn (11a) is used and $|\cos\theta_{\mathbf{xm}}| < 1/\|\mathbf{m}(t)\|$.*

It follows from the theorem that either eqn (11a) or eqn (11b) can guarantee error reduction for each learning step as long as $0 < \alpha < 1$ and thus the problems with eqns (10b, d) are avoided. At the same time, it is now possible to modify the learning speed simply by controlling one parameter α like in other learning rules, e.g., the Kohonen learning rule (Kohonen, 1988).

Table 2 and Figure 5(a) show the simulation results by using learning rule eqn (11a) in comparison with those of eqn (10b). It can be seen that the learning by eqn (11a) produced a better solution. Figures 5(b) and (c) show the learning process with different random initial values. They are quite similar to those in Figures 4(b) and (c), but with one different point here that we use a much larger learning rate. We let α start at 0.1 and linearly reduce to 0.001 at the first 500 learning steps and hereafter keep it unchanged at 0.001. If we use such a learning rate in Section 4, a rapid divergence will occur.

TABLE 2
A Comparison of Results Given by Learning Rule eqn (10b) and eqn (11a). The Solutions are Listed in the Same Way as in Table 1

	a_1	a_2
NTLS eqn (10b)		
last	0.2589	0.4832
means	0.2589	0.4833
variances	0.36×10^{-6}	0.324×10^{-5}
NTLS eqn (11a)		
last	0.2542	0.4988
means	0.2542	0.4989
variances	0.25×10^{-6}	0.324×10^{-5}

For either the improved learning rule eqn (11a) or eqn (11b), there is also one other very important advantage: they have the ability to resist the influence of outlier noise. In contrast, the least square methods perform poorly in such situations. In the following, first we show some results of computer simulations, and then we try to reveal the reasons why the improved learning has such an ability.

Instead of using the data set of Figure 3(b), we use the data set with outlier given in Figure 6(a). Table 3 and Figures 6(b), (c) and (d) show the simulation results obtained from the data set by using LS1, LS2, and NTLS with learning eqn (11a), respectively. From these results, it can be seen that although both LS1 and LS2 give quite poor solutions, our NTLS with the improved learning rule produces a much better solution indicating that the method does have a good outlier-resistant ability.

There is a basic reason behind the fact that NTLS with the improved learning equation could resist outlier disturbances. It is just the adaptive learning rate α_a: For an outlier input, the value of $|\|\mathbf{x}(t)\|^2 - z^2(t)|$ usually becomes quite large, thus α_a becomes very small, which forces the wrong modification to give the current $\mathbf{m}$ a small change. Thus, the learning could remain stable and the right solution could be retained. Theoretically, this follows from the fact that the averaged forms of eqns (11a and 11b) do not contain the pure data correlation matrix $E(\mathbf{x}\mathbf{x}^T)$ but a modified version that also depends on the form of α_a. In the modified matrix, the large magnitude deviations have a smaller weight.

A nonlinear PCA-like unit that has even better outlier resistance has been recently proposed by (Oja, Ogawa, & Wangviwattana, 1991).

In addition to eqns (11a and 11b), some other modifications on the adaptive rate α_a are possible, e.g., one workable version of α_a is

$$\alpha_a = 1/\|\mathbf{x}(t)\|^2. \quad (11c)$$

For any version of α_a given by anyone of eqns (11a, b, c,), there is also some chance that the learning can become unstable if $\|\mathbf{x}(t)\|^2 - z^2(t)$ or $\|\mathbf{x}(t)\|^2$ is near

TABLE 3
The Solutions by Different Methods On The Data Set With Outlier. The Solutions are Listed in the Same Way as in Table 1

	a_1	a_2
Original	0.25	0.5
LS1	0.2419	0.3922
LS2	0.2181	0.5797
NTLS (last)	0.2449	0.4922
NTLS (means)	0.2450	0.4917
NTLS (vars.)	0.2158×10^{-7}	0.2307×10^{-6}

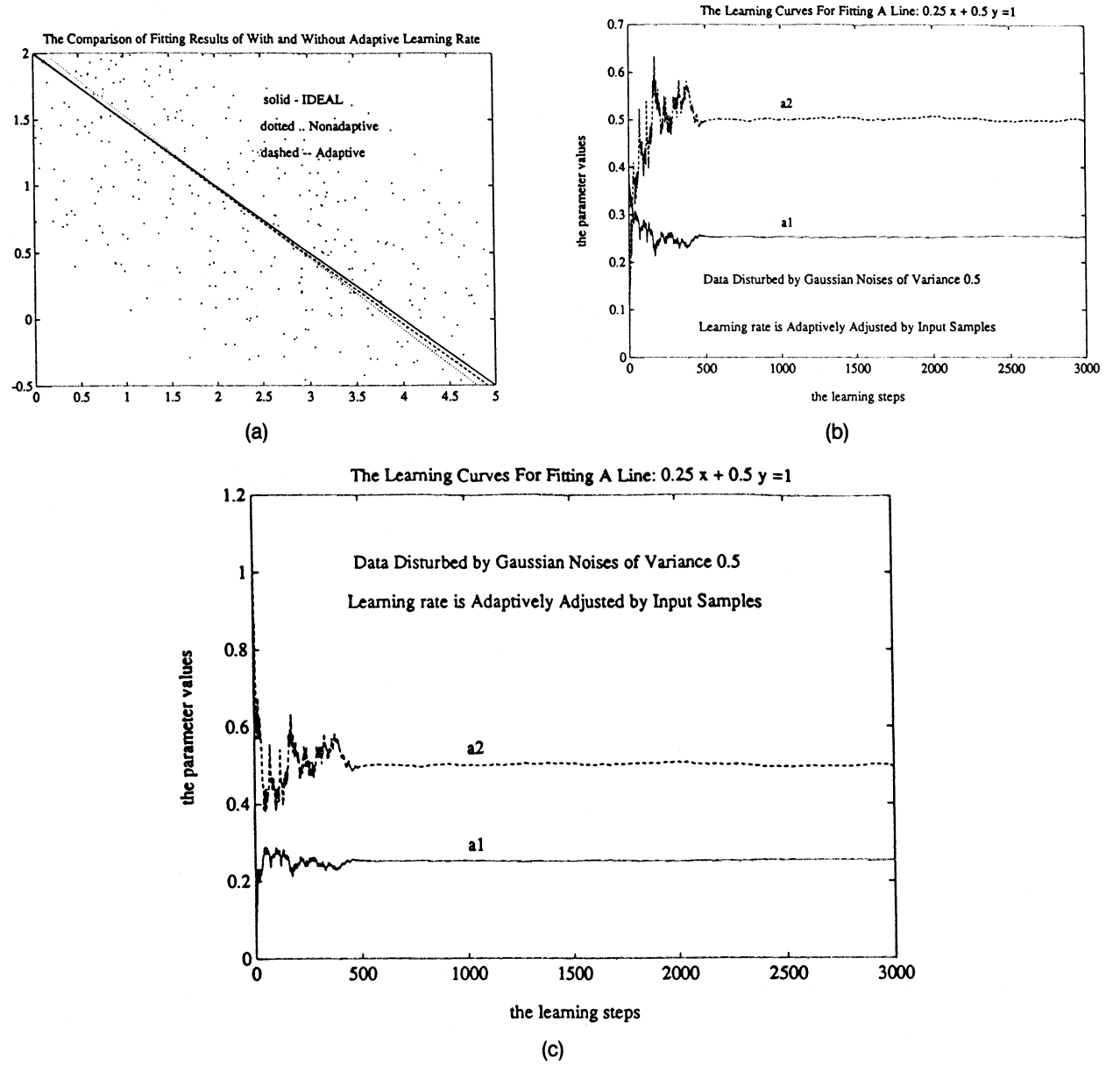


FIGURE 5. The results of line fitting by using learning equation eqn (11a) in comparison with eqn (10b). The same noisy data set D'_x shown in Figure 3(b) is still used. (a) The lines of the fitting results by using eqn (11a) and eqn (10b): the dashed line is the result of eqn (11a) which adaptively adjusts the learning rate α_a, the dotted line is the result of eqn (10b), and the solid line is the ideal (original) solution. (b) and (c) The learning processes with different random initial estimates. After 500 steps, the two processes are almost the same.

zero, although the chance is very rare in practice. One solution is to use the following α'_a to replace α_a:

$$\alpha'_a = \begin{cases} \alpha_a, & \text{if } \alpha_a \geq T; \\ T, & \text{otherwise;} \end{cases} \tag{11d}$$

where $T \geq 1$ is a predefined threshold.

6. HIGH-ORDER NEURAL UNITS FOR NONLINEAR SURFACE FITTING

High-order neural units have been advocated by Giles and Maxwell (1987). Here we show that the unit of Figure 2 can be generalized into a high-order unit for fitting nonlinear curves and hypersurface which could be expressed by eqn (6a), i.e.,

$$a_1 f_1(\mathbf{x}) + a_2 f_2(\mathbf{x}) + \ldots + a_m f_m(\mathbf{x}) + c_0 = 0,$$

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T,$$

where $f_i(\mathbf{x})$ is a function of $\mathbf{x}$. For example, for polynomial curves, $f_i(\mathbf{x})$ is a multiplicative combination of the components of $\mathbf{x}$ below

$$f_i(\mathbf{x}) = x_1^{r_1} x_2^{r_2} \ldots x_n^{r_n}, \quad \text{with integers}$$

$$r_j \geq 0, \quad j = 1, \ldots, n.$$

In the quadratic case

$$a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + a_4 x_1 + a_5 x_2 + c_0 = 0,$$

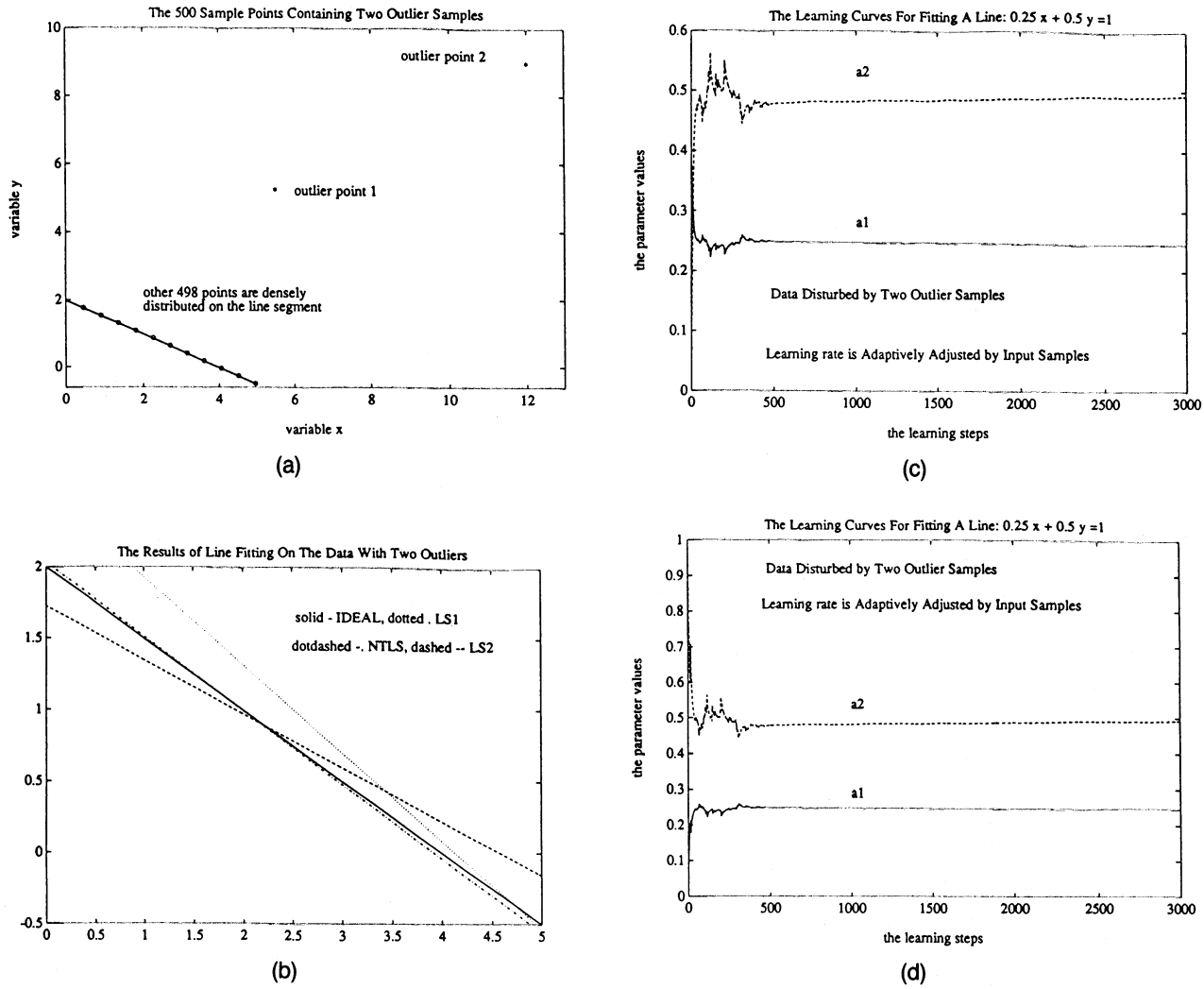


FIGURE 6. The results of line fitting on the data set with outlier. (a) Two of the pure 500 sample points given in Figure 3(a) are disturbed by two strong outlier noise points, while the other 498 points are still pure data. (b) The lines of the fitting results by LS1, LS2, and NTLS are visualized in comparison with the original line. (c) and (d) The learning processes with different random initial estimates. The learning rates are the same as those used in Figure 5(b). After about 1,000 learning steps, two learning processes give almost the same converged solutions only with very slight fluctuations.

we have

$$f_1(\mathbf{x}) = x_1^2, \quad f_2(\mathbf{x}) = x_1x_2, \quad f_3(\mathbf{x}) = x_2^2,$$
$$f_4(\mathbf{x}) = x_1, \quad f_5(\mathbf{x}) = x_2.$$

In fact, a generalized high-order unit is just like that given in Figure 2. The difference is that each weight $\mu_i(t)$ is now regarded as a high-order connection and each input is not the component of data vector $\mathbf{x}$ but its high-order combination $f_i(\mathbf{x})$.

Let $\mathbf{f} = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})]^T$. We first calculate $\mathbf{e_f} = E(\mathbf{f})$, and if $\mathbf{e_f}$ is a zero vector, then the high-order weight vector $\mathbf{m}(t) = [\mu_1(t), \ldots, \mu_m(t)]^T$ could be learned directly by eqn (8a) or eqn (8b) with $\mathbf{x}(t)$ being replaced by $\mathbf{f}$. In the general case, by using the strategy given in eqn (5a), we use $\mathbf{f}' = \mathbf{f} - \mathbf{e_f}$ to train the unit by learning eqns (8a and b) (or eqns (10b and d)) as well as their improved versions eqn (11a and b). After $\mathbf{m}$ has reached its converged value $\mathbf{m_c}$, we take the solution for parameters as $[a_1, a_2 \ldots, a_m]^T = \mathbf{m_c}$ and $c_0 = -\mathbf{e_f}^T\mathbf{m_c}$.

In the following, we give some simulation results to illustrate the performance of our neural method. A data set $D_x = \{(x_i, y_i, z_i), i = 1, \ldots, 993\}$ comes from an ellipsoid

$$0.04x^2 + 0.0625y^2 + 0.1111z^2 = 1,$$

in a way given in Figure 7(a). Similar to line fitting described in Sections 4 and 5, the problem is to use a parameterized model (e.g., $a_1x^2 + a_2y^2 + a_3z^2 = 1$) to fit D_x so that the right parameters a_1, a_2, a_3 are solved. Again the fitting problem on the pure data set is not our interest here since both the usual LS method and our neural method can give the perfect solution. Instead, we generate two noise-disturbed data sets as follows:

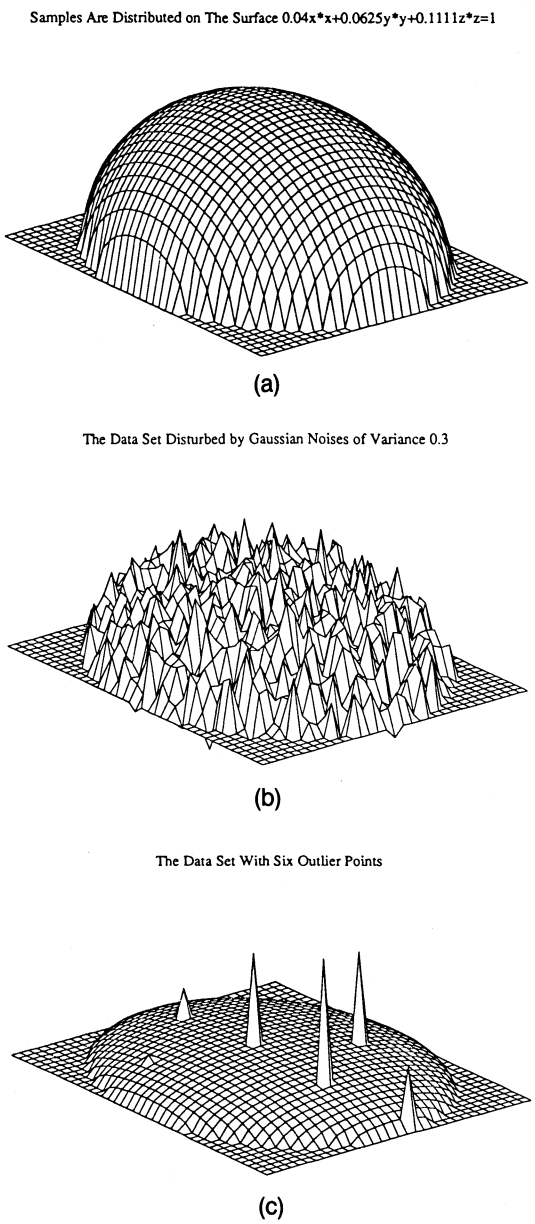


FIGURE 7. The data set used in surface fitting simulations. (a) 993 data points of our data set are obtained by sampling the ellipsoid in such a way that the sampling intervals on $x - y$ plane are uniform. Thus, each knot on the ellipsoid is a sample point. (b) These irregular knots are the 993 noise-disturbed sample points obtained by using Gaussian noise with variance $\sigma^2 = 0.3$ in the way given by eqn (12a). (c) Six of the pure 993 sample points given in (a) are disturbed by six randomly generated wild points.

1. Considering that in computer vision problems the influence of noise on variables x, y can be neglected usually, while the z variable has been contaminated by noise much stronger than x, y, we add Gaussian noise on D_x as follows

$$z'_i = z_i + n_z, \quad x'_i = x_i, \quad y'_i = y_i, \qquad i = 1, \ldots, 993 \quad (12a)$$

where $E[n_z] = 0$, $\text{var}[n_z] = \sigma^2$.

2. We let some randomly generated noise points be added to the points of the pure data set D_x.

Figure 7(b) shows the data set generated from eqn (12a) with noise variance $\sigma^2 = 0.3$. Figure 7(c) is the result of using six randomly generated wild points to disturb the pure data set D_x. We denote both of the above two noisy data sets by $D'_x = \{(x'_i, y'_i, z'_i), i = 1, \ldots, 993\}$. In the following, our simulations are conducted on the two noisy sets by using LS method and our NTLS, respectively.

The LS method is implemented by parameterization $a_1x^2 + a_2y^2 + a_3z^2 = 1$, and the solution $\mathbf{a} = [a_1, a_2, a_3]^T$ is obtained by

$$\mathbf{a} = (X^TX)^{-1}X^T\mathbf{b}, \quad \mathbf{b} = [1, \ldots, 1]^T$$

$$X^T = \begin{pmatrix} x'_1 & \cdots & x'_{993} \\ y'_1 & \cdots & y'_{993} \\ z'_1 & \cdots & z'_{993} \end{pmatrix}.$$

The NTLS is implemented by the improved learning rule eqn (11a), similar to that in Section 4. It consists of the following steps:

1. Compute means $\mathbf{e_f} = [e_1, e_2, e_3]^T$, $e_1 = \frac{1}{993}\sum_{i=1}^{993} x'^2_i$, $e_2 = \frac{1}{993}\sum_{i=1}^{993} y'^2_i$ and $e_3 = \frac{1}{993}\sum_{i=1}^{993} z'^2_i$.
2. Subtract the means from each data point, i.e., $f_1^i = x'^2_i - e_1$, $f_2^{(i)} = y'^2_i - e_2$ and $f_3^{(i)} = z'^2_i - e_3$.
3. Initialize weight vector $\mathbf{m} = [\mu_1, \mu_2, \mu_3]^T$ by a random number from a uniform distribution on $[0, 1] \times [0, 1]$, and let $\mathbf{f} = [f_1^{(j)}, f_2^{(j)}, f_3^{(j)}]^T$, where j is a random integer, with equal probability any integer of $\{1, 2, \ldots, 993\}$.
4. Adapt $\mathbf{m}$ by

$$\mathbf{m}(t+1) = \mathbf{m}(t) - \alpha\alpha_a z(t)[\mathbf{f} - z(t)\mathbf{m}(t)], \quad (12b)$$

where $z(t) = \mathbf{m}(t)^T\mathbf{f}$, $\alpha_a = 1/|\|\mathbf{f}\|^2 - z^2(t)|$, and α starts at 0.1 and linearly reduces to 0.001 at the first 500 learning steps, after which it remains unchanged at 0.001.
5. Calculate the current solution $\mathbf{a}(t+1) = [a_1(t+1), a_2(t+1), a_3(t+1)]^T$ through transforming equation $[x^2, y^2, z^2]\mathbf{m} + c_0 = 0$ into $a_1(t+1)x^2 + a_2(t+1)y^2 + a_3(t+1)z^2 = 1$, i.e.,

$$a_1(t+1) = -\mu_1(t+1)/c_0,$$

$$a_2(t+1) = -\mu_2(t+1)/c_0,$$

$$a_3(t+1) = -\mu_3(t+1)/c_0, \quad c_0 = \mathbf{m}(t+1)^T\mathbf{e}_f$$

Table 4 and Figures 8(a)–(f) show the results of the experiment on the data set disturbed by strong Gaussian noise (see Figure 7(b)). Table 4 already shows that the solution of NTLS is better than that of LS, and Figures 8(a)–(f) show visibly that in all the three directions NTLS has obtained better results than the LS method. Especially in the direction vertical to x and y axis, respectively, NTLS outperform LS significantly.

Similarly, Table 5 and Figures 9(a)–(f) show the results of the experiment on the data set disturbed by

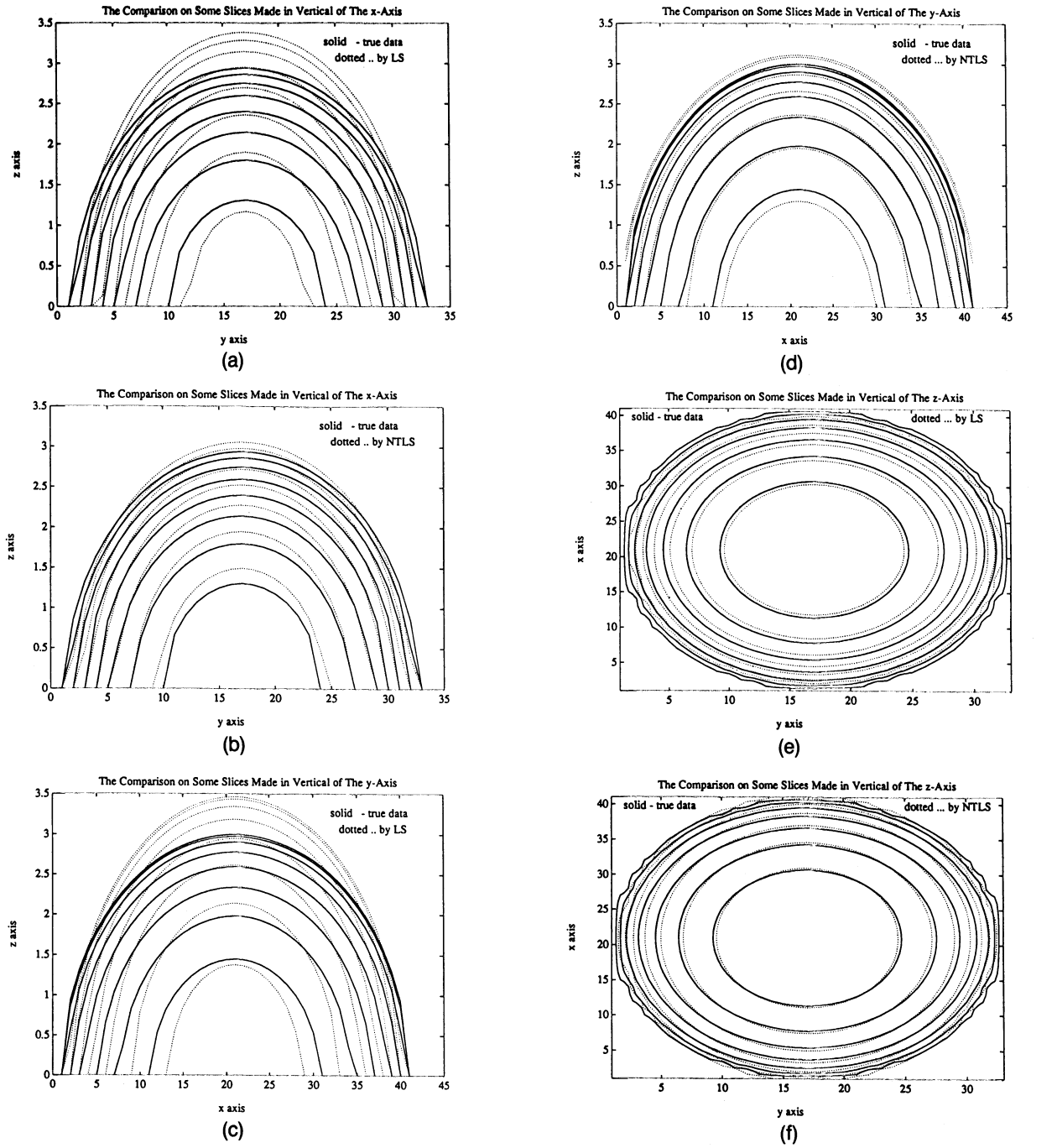


FIGURE 8. The results of surface fitting on the data set given in Figure 7(b). To visualize the estimated ellipsoids by LS and NTLS, we show some slices obtained by intersecting them along each of the directions vertical to x, y, z axis. (a), (c), and (e) are those slices of the result by NTLS intersected along the direction vertical to x, y, z axis, respectively, (b), (d), and (f) give those counterparts of the result by LS.

TABLE 4
The Solutions by LS and NTLS On Surface Fitting With Gaussian Noise. The Solutions are Listed also in the Same Way as in Table 1

	a_1	a_2	a_3
original	0.04	0.0625	0.1111
LS	0.0438	0.0686	0.0832
NTLS (last)	0.0383	0.0677	0.1023
NTLS (means)	0.038	0.0673	0.1029
NTLS (vars.)	0.115×10^{-7}	0.1544×10^{-6}	0.1622×10^{-6}

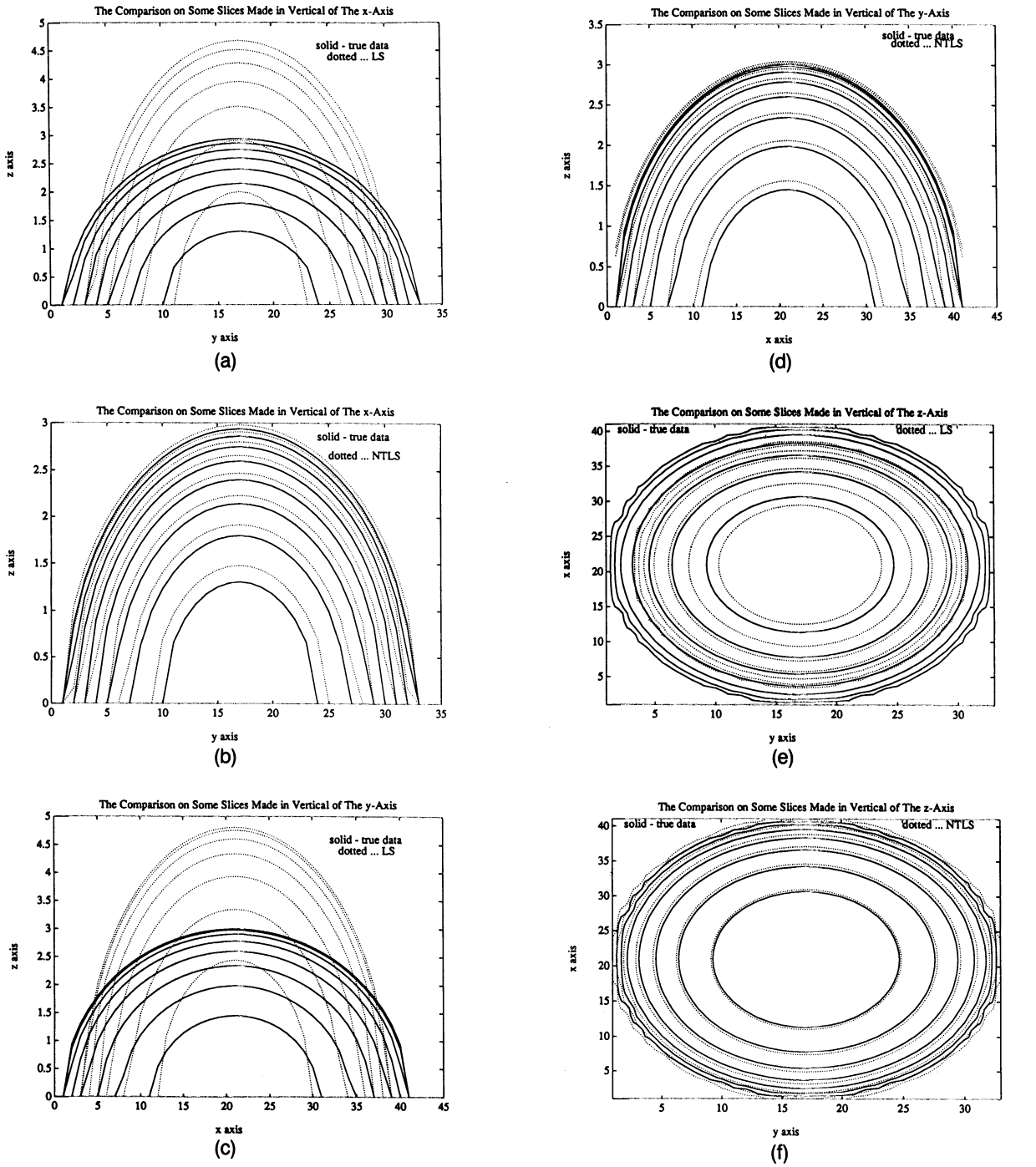
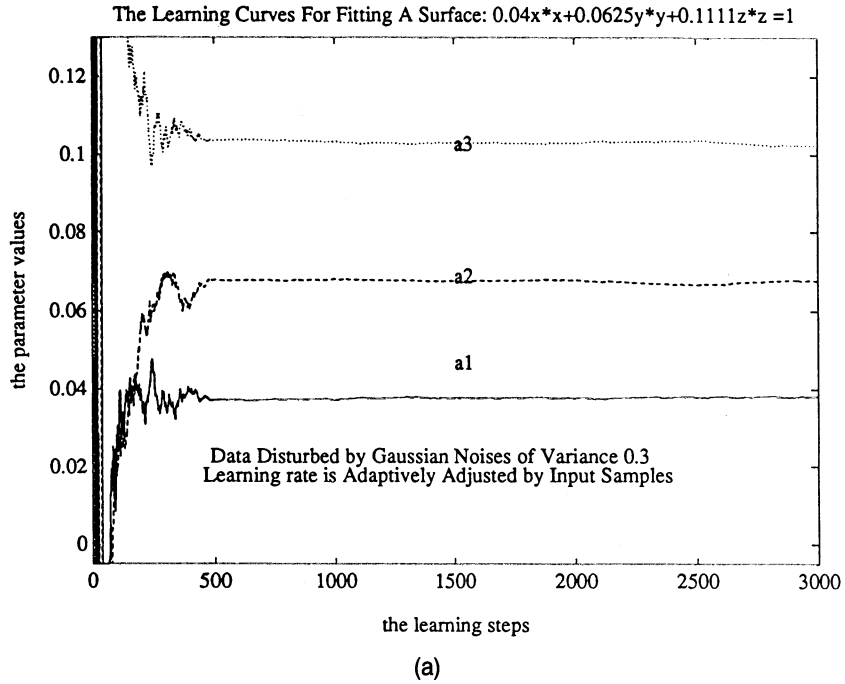


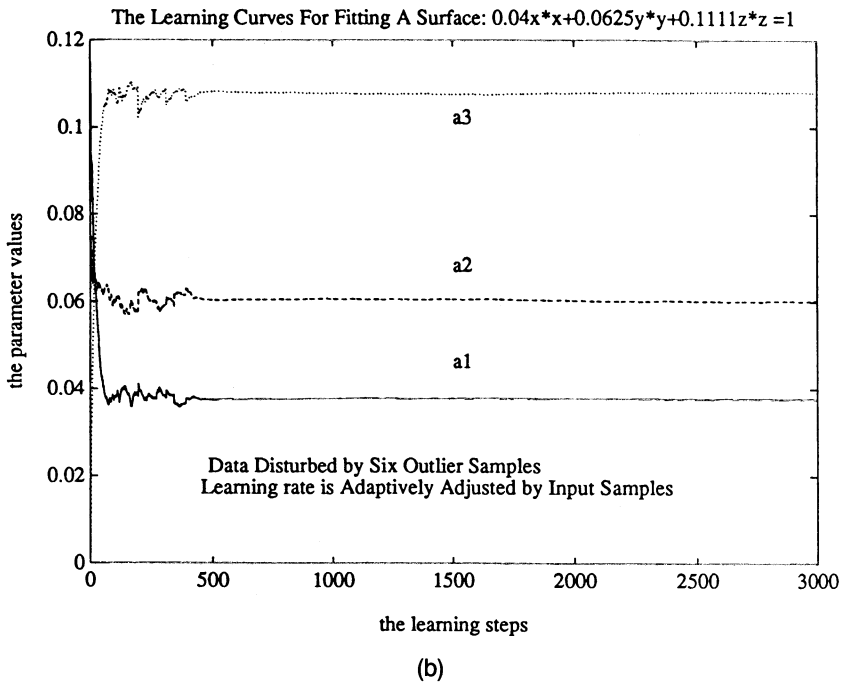
FIGURE 9. The results of surface fitting on the data set given in Figure 7(c). Here (a), (b), (c), (d), (e), and (f) are, respectively, the counterparts of Figs. 8(a), (b), (c), (d), (e), and (f).

TABLE 5
The Solutions Obtained by LS and NTLS On Surface Fitting With Outlier. The Solutions are Listed Again in the Same Way as in Table 1

	a_1	a_2	a_3
original	0.04	0.0625	0.1111
LS	0.0516	0.0825	0.0432
NTLS (last)	0.0376	0.0603	0.1081
NTLS (means)	0.0377	0.0602	0.1081
NTLS (vars.)	0.592×10^{-9}	0.3428×10^{-8}	0.214×10^{-8}



(a)



(b)

FIGURE 10. The learning processes of NTLS: (a) on the data set of Figure 7(b), some strong fluctuations could be observed at the earlier steps, but they rapidly reduced within the first 100 steps. (b) on the data set of Figure 7(c), a rapid convergence is observed.

outlier (see Figure 7(c)). From both Table 5 and Figures 9(a), (c), and (e), we see that the results by LS are very poor in all the three directions due to the reason that the least square method cannot resist outlier disturbances. However, it follows from Table 5 and Figures 9(b), (d), and (f) that the estimated solution by NTLS is quite good. It is obvious that NTLS can significantly outperform LS in all the three directions. This again verified that NTLS has the ability of resisting outlier disturbances.

Figure 10(a) and (b) show the learning process of NTLS on the data set of Figure 7(b) and Figure 7(c), respectively. We again see the advantage of the improved learning rule through two obvious facts: (a) the learning exhibited in Figure 10(a) can still stabilize rapidly although at the earlier steps all the three weights oscillated

strongly; (b) the rapid convergence of the learning depicted in Figure 10(b) is obtained by a large learning rate while the learning still remained stable.

7. CONCLUSIONS

For data or pattern representation, it is conventionally regarded that principal components are important while minor components are mostly noise. However, this paper has shown that cases exist in which the minor components have the same importance as principal components. We have shown that the fitting problems could be solved in the TLS sense by extracting the minor component of data set and that a linear neural unit with modified anti-Hebbian learning can adaptively fulfill such a task. In other words, a simple neural unit could act as an optimal fitting analyzer which can outperform the least square method significantly. The results of computer simulations demonstrated that our method can provide a new tool for adaptively solving the classical optimal fitting (or modelling) problem.

REFERENCES

Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks,* **2,** 52–58.

Baldi, P., & Hornik, K. (1991). Back-propagation and unsupervised learning in linear networks. In Y. Chauvin and D. E. Rumelhart (Eds.), *Back propagation: Theory, architectures and applications.* Erlbaum Associates.

Chauvin, Y. (1989). Principal component analysis by gradient descent on a constrained linear Hebbian cell. *Proceedings of IEEE International Conference on Neural Networks,* (Vol I, pp. 373–380). New York: IEEE Press.

Foldiak, P. (1989). Adaptive network for optimal linear feature extraction. *Proceedings of IEEE International Conference on Neural networks,* Washington D.C. (Vol I, pp. 401–405). New York: IEEE Press.

Giles, C. L., & Maxwell, T. (1987). Learning, invariance, and generalization in high order neural networks. *Applied Optics,* **26,** 4972–4978.

Golub, G., & Van Loan, C. (1983). *Matrix computations.* Baltimore, MD: Johns Hopkins University Press.

Karhunen, J. (1982). On the recursive estimation of the eigenvectors of correlation type matrices. Lic. Tech. Thesis, Helsinki University of Technology.

Kohonen, T. (1988). *Self-organization and associative memory.* Berlin: Springer-Verlag.

Kung, S. Y. (1990). Constrained principal component analysis via an orthogonal learning network. *Proceedings of 1990 IEEE International Symposium on Circuits and Systems,* New Orleans, LA (Vol. 1, pp. 138–140). New York: IEEE Press.

Kung, S. Y., & Diamantaras, K. I. (1990). A neural networks learning algorithm for adaptive principal component extraction (APEX). *Proceedings of* 1990 *IEEE ASSP Conference* (pp. 861–864). New York: IEEE Press.

Kushner, H., & Clark, D. (1978). *Stochastic approximation methods for constrained and unconstrained systems.* New York: Springer-Verlag.

Linsker, E. (1986a). From basic network principles to neural architecture: Emergence of spatial opponent cells. *Proceedings of the National Academy of Sciences USA,* **83,** 7508–7512.

Linsker, E. (1986b). From basic network principles to neural architecture: Emergence of orientation selective cells. *Proceedings of the National Academy of Sciences USA,* **83,** 8390–8394.

Linsker, E. (1986c). From basic network principles to neural architecture: Emergence of orientation columns. *Proceedings of the National Academy of Sciences USA,* **83,** 8779–8783.

Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE Transactions of Automatic Control,* **22,** 551–575.

Oja, E. (1982). A simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology,* **16,** 267–273.

Oja, E. (1989). Neural networks, principal components, and subspaces. *International Journal of Neural Systems,* **1,** 61–68.

Oja, E., & Karhunen, J. (1985). On stochastic approximation of eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematics Analysis and Applications,* **106,** 69–84.

Oja, E., Ogawa, H., & Wangviwattana, J. (1991). Learning in nonlinear constrained Hebbian networks. *Proceedings of the International Conference on Artificial Neural Networks.* Helsinki, Finland (pp. 385–390) Amsterdam: North-Holland.

Pearson, K. (1901). On lines and planes of closest fit to points in space. *Phil. Mag.,* **2,** 559–572.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Ann. Math. Statist.,* **22,** 400–407.

Rubner, J., & Tavan, P. (1989). A self-organizing network for principal-component analysis. *Europhysics Letters,* **10,** 693–689.

Rubner, J., & Schulten, K. (1990). Development of feature detectors by self-organization. *Biological Cybernetics,* **62,** 193–199.

Sanger, T. D. (1980). Optimal unsupervised learning in a single-layer linear feed forward neural network. *Neural Networks,* **2,** 459–473.

APPENDIX

1. The Proof of Part (1) of Theorem 1

Following the technique used by one of the authors to prove Lemma 2 in (Oja & Karhunen, 1985, p. 73), let $\mathbf{c}_i$, λ_i denote the unit eigenvectors and eigenvalues of matrix R and assume that the smallest eigenvalue λ_n is strictly smaller than the other eigenvalues. Express the solution $\mathbf{m}(t)$ as

$$\mathbf{m}(t) = \sum_{i=1}^{n} \eta_i(t)\mathbf{c}_i, \tag{A1}$$

which is always possible since the eigenvectors of a symmetrical matrix form an orthonormal basis. Then (8e) yields

$$d\eta_i/dt = -\lambda_i\eta_i + (\mathbf{m}(t)^T R\mathbf{m}(t))\eta_i. \tag{A2}$$

If $\eta_n(0) = 0$, then $\eta_n(t)$ must be zero for all t; this implies that if $\eta_n(0) \neq 0$, then $\eta_n(t)$ is nonzero for all t. Therefore, we may define $\zeta_i(t) = \eta_i(t)/\eta_n(t)$ and get directly from (A2) the equation

$$d\zeta_i(t)/dt = (\lambda_n - \lambda_i)\zeta_i(t), \tag{A3}$$

which shows that all $\zeta_i(t)$, $i \neq n$, tend to zero as $t \to \infty$.

Therefore, asymptotically

$$\mathbf{m}(t) = \eta_n(t)\mathbf{c}_n, \tag{A4}$$

and for $\eta_n(t)$, eqn (A2) yields

$$d\eta_n(t)/dt = \lambda_n(\eta_n^3(t) - \eta_n(t)). \tag{A5}$$

This shows directly that if $\lambda_n = 0$, then $\eta_n(t) = \eta_n(0)$ for all t, and $m(t)$ tends to $\eta_n(0)c_n = (\mathbf{m}(0)^T c_n)c_n$. .QED.

(Note: If $0 < \lambda_n$, eqn (A5) also shows that $|\eta_n(t)|$ tends to either zero (if $|\eta_n(0)| < 1$) or infinity (if $|\eta_n(0)| > 1$), and the point $\eta_n(t) = 1$ is a unstable fixed point.)

2. The Proof of Part (2) of Theorem 1

Let $A = -R$ and substitute it into eqn (8f), which gives eqn (11) of (Oja & Karhunen, 1985, p. 73). Because A is a symmetric matrix, the conclusion of Theorem 1 (Oja & Karhunen, 1985, p. 73) directly applies. .QED.

3. The Proof of Proposition 1

For $r' = |\mathbf{m}(t+1)^T\mathbf{x}(t)|^2/\|\mathbf{m}(t+1)\|^2$ and $r = |\mathbf{m}(t)^T\mathbf{x}(t)|^2/\|\mathbf{m}(t)\|^2$, the task is to show $\rho(\alpha) = r'/r > 1$ under the conditions that

$$\alpha > \alpha_b = \frac{2}{\|\mathbf{x}(t)\|^2(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}})} \quad \text{and}$$

$$|\cos\theta_{\mathbf{xm}}| < \frac{1}{\sqrt{2}\|\mathbf{m}(t)\|}.$$

By eqn (10b), we have

$$\mathbf{m}(t+1)^T\mathbf{x}(t) = z(t)[1 - \alpha(\|\mathbf{x}(t)\|^2 - z^2(t))]$$

$$\|\mathbf{m}(t+1)\|^2 = \|\mathbf{m}(t)\|^2 - 2\alpha z^2(t)(1 - \|\mathbf{m}(t)\|^2) + \alpha^2 z^2(t)[\|\mathbf{x}(t)\|^2 - 2z^2(t) + z^2(t)\|\mathbf{m}(t)\|^2].$$

Denote

$$q = \|\mathbf{x}(t)\|^2 - z^2(t), \quad p = \|\mathbf{m}(t)\|^2, \quad u = z^2(t), \tag{A6}$$

and we can rewrite $\rho(\alpha)$ into

$$\rho(\alpha) = \frac{p(1 - \alpha q)^2}{p - 2\alpha u(1 - p) + \alpha^2 u(q - u + up)}. \tag{A7}$$

Therefore

$$\rho(\alpha) > 1 \Leftrightarrow p(1 - \alpha q)^2 > p - 2\alpha u(1 - p) + \alpha^2 u(q - u + up)$$

$$\Leftrightarrow \alpha^2(q^2 p - uq + u^2 - u^2 p) - 2\alpha(pq - u + up) > 0$$

$$\Leftrightarrow \alpha\left(\alpha - 2\frac{pq - u + up}{q^2 p - uq + u^2 - u^2 p}\right) \times (q^2 p - uq + u^2 - u^2 p) > 0. \tag{A8}$$

By substituting eqn (A6) into eqn. (A8) and noticing that

$$z(t) = \mathbf{m}(t)^T\mathbf{x}(t) = \|\mathbf{x}(t)\|\|\mathbf{m}(t)\|\cos\theta_{\mathbf{xm}},$$

we see that $\rho(\alpha) > 1$ is equivalent to

$$\alpha\left[\alpha - \frac{2}{\|\mathbf{x}(t)\|^2(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}})}\right] \times \|\mathbf{x}(t)\|^4\|\mathbf{m}(t)\|^2\sin^2\theta_{\mathbf{xm}}(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}}) > 0. \tag{A9}$$

Since α takes values in the interval $(0, 1)$, we see that this inequality holds only when

$$\alpha > \alpha_b = \frac{2}{\|\mathbf{x}(t)\|^2(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}})} \quad \text{and}$$

$$(1 - 2\|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}}) > 0.$$

Therefore, from eqn (A9), it follows that

$$\rho(\alpha) > 1 \Leftrightarrow \alpha > \alpha_b \quad \text{and} \quad |\cos\theta_{\mathbf{xm}}| < \frac{1}{2\|\mathbf{m}(t)\|}.$$

.QED.

4. The Proof of Proposition 2

By eqn (10d), we now have

$$\mathbf{m}(t+1)^T\mathbf{x}(t) = z(t)\left[1 - \alpha\left(\|\mathbf{x}(t)\|^2 - \frac{z^2(t)}{\|\mathbf{m}(t)\|^2}\right)\right]$$

$$\|\mathbf{m}(t+1)\|^2 = \|\mathbf{m}(t)\|^2 + \alpha^2 z^2(t)\Delta^2,$$

where $\Delta^2 = \|\mathbf{x}(t) - z(t)\mathbf{m}(t)/\|\mathbf{m}(t)\|^2\|^2$. If we denote

$$q = \left(\|\mathbf{x}(t)\|^2 - \frac{z^2(t)}{\|\mathbf{m}(t)\|^2}\right), \quad p = \|\mathbf{m}(t)\|^2, \quad u = z^2(t), \tag{A10}$$

then from $r' = |\mathbf{m}(t+1)^T\mathbf{x}(t)|^2/\|\mathbf{m}(t+1)\|^2$ and $r = |\mathbf{m}(t)^T\mathbf{x}(t)|^2/\|\mathbf{m}(t)\|^2$, we have

$$\rho(\alpha) = \frac{p(1 - \alpha q)^2}{p + \alpha^2 u\Delta^2}. \tag{A11}$$

Therefore

$$\rho(\alpha) > 1 \Leftrightarrow p(1 - \alpha q)^2 > p + \alpha^2 u\Delta^2$$

$$\Leftrightarrow \alpha^2(q^2 p - u\Delta^2) - 2\alpha pq > 0$$

$$\Leftrightarrow \alpha > \frac{2pq}{(q^2 p - u\Delta^2)}$$

(since the learning rate α is positive). (A12)

Denote $\alpha_b = \alpha_b(\mathbf{x}(t), \mathbf{m}(t)) = 2pq/(q^2 p - u\Delta^2)$. Next we show that it is possible for $\alpha_b < 0$ or $0 < \alpha_b < \gamma < 1$.

First, it follows from eqn (A10) that

$$pq = \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2 - z^2(t) = \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2\sin^2\theta_{\mathbf{xm}} > 0.$$

We see that $\alpha_b < 0$ is equivalent to $A(q, p, u) < 0$ with the notation $A(q, p, u) = q^2 p - u\Delta^2$. Further recalling that

$$\Delta^2 = \left\|\mathbf{x}(t) - \frac{z(t)\mathbf{m}(t)}{\|\mathbf{m}(t)\|^2}\right\|^2 = \|\mathbf{x}(t)\| - 2\frac{z^2(t)}{\|\mathbf{m}(t)\|^2} + z^2(t) = q - u/p + u,$$

we have $A(q, p, u) = q^2 p - uq + u^2(1/p - 1)$, and due to $p = \|\mathbf{m}(t)\|^2 > 0$, it is a concave parabola with respect to q. It is easy to show that when

$$p = \|\mathbf{m}(t)\|^2 > \tfrac{3}{4}, \tag{A13}$$

the equation $A(q, p, u) = 0$ has two roots $q_1 = u(1 + \sqrt{4p - 3})/2p$ and $q_2 = u(1 - \sqrt{4p - 3})/2p$, thus $A(q, p, u) < 0$ when

$$q_2 < q = \|\mathbf{x}(t)\|^2 - \frac{z^2(t)}{\|\mathbf{m}(t)\|^2} = \|\mathbf{x}(t)\|^2\sin^2\theta_{\mathbf{xm}} < q_1. \tag{A14}$$

Second, $0 < \alpha_b < \gamma < 1$ is equivalent to $A(q, p, u) > 0$ and $B(q, p, u) > 0$, where $B(q, p, u) = \gamma A(q, p, u) - 2qp = \gamma pq^2 - (\gamma u + 2p)q - \gamma u^2(1/p - 1)$. Again, $B(q, p, u)$ is a concave parabola with respect to q. If $B(q, p, u) = 0$ has no root, any q value makes $B(q, p, u) > 0$; if $B(q, p, u) = 0$ has two roots $q_1 \le q_2$, $B(q, p, u) > 0$ when either

$$q < q_1 \quad \text{or} \quad q > q_2. \tag{A15}$$

Because either eqns (A13, A14) or eqn (A15) could be true for an arbitrary input sample $\mathbf{x}(t)$, we have proved that it is possible for $\alpha_b < 0$ or $0 < \alpha_b < \gamma < 1$, and together with eqn. (A12), we have proved Proposition 2. .QED.

5. The Proof of Theorem 2

First we prove the case of using eqn (11a). Since the only difference between eqn (10b) and eqn (11a) is that α in eqn (10b) is replaced by $\alpha\alpha_a$ in eqn (11a), eqn (A7) yields the form

$$\rho(\alpha) = \frac{p(1 - \alpha\alpha_a q)^2}{p - 2\alpha\alpha_a u(1 - p) + \alpha^2\alpha_a^2 u(q - u + up)}. \tag{A16}$$

Its derivative is given by

$$\rho'(\alpha) = A(\alpha)/B(\alpha), \quad B(\alpha) = [p - 2\alpha\alpha_a u(1 - p) + \alpha^2\alpha_a^2 u(q - u + up)]^2 > 0$$

$$A(\alpha) = -2\alpha_a p(1 - \alpha\alpha_a q)C(\alpha) = -2\|\mathbf{m}(t)\|^2(1 - \alpha\alpha_a q)C(\alpha)$$

$$C(\alpha) = q[p - 2\alpha\alpha_a u(1 - p)] + [-u(1 - p) + \alpha\alpha_a u(q - u + up)] + \alpha\alpha_a qu(1 - p). \tag{A17}$$

$C(\alpha)$ is a monotonically increasing function with respect to α since

$$C'(\alpha) = \alpha_a[-2u(1 - p)q + u(q - u + up) + qu(1 - p)] = \alpha_a(upq - u^2 + u^2 p),$$

and from eqn (A6) and eqn (11a) it follows that

$$C'(\alpha) = \alpha_a z^2(t)(\|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2 - z^2(t))$$
$$= \alpha\, z^2(t)\|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2\sin^2\theta$$

Furthermore, due to

$$C(\alpha = 0) = pq - u(1-p) = \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2$$
$$- z^2(t) = \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2\sin^2\theta_{\mathbf{xm}} > 0,$$

we can also see that $C(\alpha) > 0$ for any $\alpha \geq 0$.

In addition, from eqn (11a) and eqn (A6), we have $\alpha_a = |1/q|$. It follows from eqn (A17) that if $q > 0$, then

$$A(\alpha) = -2\alpha_a\|\mathbf{m}(t)\|^2(1-\alpha)C(\alpha),$$

and if $q < 0$, then

$$A(\alpha) = -2\alpha_a\|\mathbf{m}(t)\|^2(1+\alpha)C(\alpha).$$

In both cases, $A(\alpha) < 0$ (and thus $\rho'(\alpha) < 0$) when $\alpha \in (0, 1)$ (i.e., $\rho(\alpha)$ is a monotonic decreasing function of α on $[0, 1]$).

Specifically, if $q = \|\mathbf{x}(t)\|^2(1 - \|\mathbf{m}(t)\|^2\cos^2\theta_{\mathbf{xm}}) > 0$ (i.e., $|\cos\theta_{\mathbf{xm}}| < 1/\|\mathbf{m}(t)\|$), then by eqn (A16) we have $\rho(\alpha = 0) = 1$ and $\rho(\alpha = 1) = 0$.

Second we prove the case of using eqn (11b). Similarly, replacing α in eqn (A11) by $\alpha\alpha_a$, we have for eqn (11b)

$$\rho(\alpha) = \frac{p(1-\alpha\alpha_a q)^2}{p + \alpha^2\alpha_a^2 u\Delta^2}. \tag{A18}$$

Its derivative is now given by

$$\rho'(\alpha) = A(\alpha)/B(\alpha), \quad B(\alpha) = (p + \alpha^2\alpha_a^2 u\Delta^2)^2 \geq 0$$
$$A(\alpha) = -2\alpha_a\|\mathbf{m}(t)\|^2(1-\alpha\alpha_a q)C(\alpha)$$
$$C(\alpha) = q[p + \alpha^2\alpha_a^2 u\Delta^2] + 2u\Delta^2\alpha\alpha_a(1-\alpha\alpha_a q). \tag{A19}$$

We further have

$$C'(\alpha) = 2u\Delta^2\alpha_a(1-\alpha\alpha_a q).$$

In addition, from eqn (11b) and eqn (A10), we have $\alpha_a = 1/q$, thus

$$C'(\alpha) = 2u\Delta^2 1/q(1-\alpha) > 0,$$

when $\alpha \in (0, 1)$ because $u > 0$, $q > 0$, $\Delta^2 > 0$ according to eqn (A10). Therefore, in the interval $[0, 1]$, $C(\alpha)$ is a monotonically increasing function with respect to α. Due to

$$C(\alpha = 0) = pq = \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)^2 - z^2(t)$$
$$= \|\mathbf{x}(t)\|^2\|\mathbf{m}(t)\|^2\sin^2\theta_{\mathbf{xm}} > 0$$

we can also infer that $C(\alpha) > 0$ for any $\alpha \in [0, 1]$.

Furthermore, by putting $\alpha_a = 1/q$ into $A(\alpha)$, we have

$$A(\alpha) = -2\alpha_a\|\mathbf{m}(t)\|^2(1-\alpha)C(\alpha) < 0,$$

when $\alpha \in [0, 1]$(i.e., $\rho(\alpha)$ is a monotonic decreasing function of α on $[0, 1]$), and by eqn (A18) we have $\rho(\alpha = 0) = 1$ and $\rho(\alpha = 1) = 0$. .QED.

6. An Additional Analysis on eqn (11a)

By taking the conditional expectation of the right hand side of the equation over x, keeping $\mathbf{m}(t)$ fixed, we can get the corresponding averaged continuous-time equation:

$$d\mathbf{m}(t)/dt = -R(\mathbf{m}(t))\mathbf{m}(t)$$
$$+ [\mathbf{m}(t)^T R(\mathbf{m}(t))\mathbf{m}(t)]\mathbf{m}(t), \tag{A20}$$

where

$$R(\mathbf{m}(t)) = E\{\alpha_a(\mathbf{x}, \mathbf{m}(t))\mathbf{x}\mathbf{x}^T\mathbf{m}(t)\}. \tag{A21}$$

Assume that $\alpha_a(\mathbf{x}, \mathbf{m}(t)) > 0$ for all $\mathbf{m}(t)$, and

$$\lim_{m(t)\to\mathbf{0}} \alpha_a(\mathbf{x}, \mathbf{m}(t)) = 1/\mathbf{x}^T\mathbf{x}. \tag{A22}$$

(Note: these conditions on α_a are valid for the cases of eqns (11a), (11c), and (11d).)

Then it further holds that $R(\mathbf{m}(t))$ tends to $R_1 = E\{\mathbf{x}\mathbf{x}^T/\mathbf{x}^T\mathbf{x}\}$ if $\mathbf{m}(t) \to 0$.

Now eqn (A20) gives $\frac{1}{2}d\|\mathbf{m}(t)\|^2/dt = \mathbf{m}(t)^T R(\mathbf{m}(t))\mathbf{m}(t)$ ($\|\mathbf{m}(t)\|^2 - 1$). Due to [A21] it holds that $\mathbf{m}(t)^T R(\mathbf{m}(t))\mathbf{m}(t)$ is positive for all $\mathbf{m}(t) \neq \mathbf{0}$, and it follows that if $\|\mathbf{m}(0)\| < 1$, then $\|\mathbf{m}(t)\|$ will tend to zero as $t \to \infty$. This in turn implies that $R(\mathbf{m}(t))$ will tend to R_1 and asymptotically eqn (A20) becomes

$$d\mathbf{m}(t)/dt = -R_1\mathbf{m}(t) + [\mathbf{m}(t)^T R_1\mathbf{m}(t)]\mathbf{m}(t), \tag{A23}$$

which can be analyzed exactly in the same way as that used in this Appendix for proving Part (1) of Theorem 1. We then know that $\mathbf{m}(t)$ will asymptotically have the direction of the minor eigenvector of matrix R_1.