Robust Principal Component Analysis by Self-Organizing Rules Based on Statistical Physics Approach

Lei Xu, *Senior Member, IEEE*, and Alan L. Yuille, *Member, IEEE*

***Abstract*—This paper applies statistical physics to the problem of robust principal component analysis (PCA). The commonly used PCA learning rules are first related to energy functions. These functions are generalized by adding a binary decision field with a given prior distribution so that outliers in the data are dealt with explicitly in order to make PCA robust. Each of the generalized energy functions is then used to define a Gibbs distribution from which a marginal distribution is obtained by summing over the binary decision field. The marginal distribution defines an effective energy function, from which self-organizing rules have been developed for robust PCA. Under the presence of outliers, both the standard PCA methods and the existing self-organizing PCA rules studied in the literature of neural networks perform quite poorly. By contrast, the robust rules proposed here resist outliers well and perform excellently for fulfilling various PCA-like tasks such as obtaining the first principal component vector, the first k principal component vectors, and directly finding the subspace spanned by the first k vector principal component vectors without solving for each vector individually. Comparative experiments have been made, and the results show that our robust rules improve the performances of the existing PCA algorithms significantly when outliers are present.**

I. Introduction

PRINCIPAL component analysis (PCA) is an essential technique for data compression and feature extraction, and has been widely used in statistical data analysis, communication theory, pattern recognition, and image processing.

Oja [22] found that a simple linear neuron with a constrained Hebbian learning rule can extract the principal component from stationary input data, thereby building the first connection between self-organizing rules in neural networks and PCA techniques. Thereafter, there was increasing interest in the study of connections between PCA and neural networks. A symmetrical error-correcting learning rule was proposed by Williams [35] for a two-layer network which can discover the subspace spanned by the first k principal components. Multilayer perceptron (MLP) neural networks, which learn by the backpropagation algorithm in *supervised autoassociative* mode, have been suggested for data compression [8] and have been shown to be closely connected to PCA [1], [5]. A number of unsupervised learning rules for extracting principal components or their spanned subspace have been also proposed and studied [16], [26], [36], [28], [34], [32], [20], [4], [6]. The relationships between PCA and the emergence of feature-analyzing properties in the cortex field of biological systems have been revealed and studied [18], [3], [40], [21].

Furthermore, some extensions of PCA have also been made. For example, in [17], the original PCA problem has been extended into the so-called *constrained principal components analysis*, i.e., the principal components of data are extracted in such a way that they are constrained to be orthogonal to some undesired subspace (e.g., a subspace dominated by interfering noise or redundant components). In [37], [27], instead of extracting principal components or the subspace spanned by the principal components, the rules for extracting *minor components* (i.e., the counterparts of the *principal components*) have been investigated, and have been applied to total least square curve and surface fitting [37] and dual subspace pattern recognition [38].

However, almost all the above-mentioned PCA algorithms are based on the assumption that data have not been spoiled by outliers.[1] In practice, real data often contain some outliers, and usually they are not easy to separate from the data set. As will be shown by the experiments given in this paper, these outliers will significantly deteriorate the performances of the existing PCA algorithms. Currently, little attention has been paid to this problem in neural network literature, although the problem is essentially important for real applications. In the literature of statistics, the importance of this problem is well appreciated, and a number of efforts have been made to tackle the problem. As will be further shown in Section VI, however, the existing algorithms in the statistics literature [14], [33], [10], [9] are based on computations in *batch way*, and are obviously different from those self-organizing or adaptive rules currently studied in the paradigm of neural network learning.

This paper attempts to develop self-organizing rules for robust PCA. Recently, there have been a number of successes in applying the statistical physics approach to a variety of computer vision problems [42], [43], [41], [12]. It has also been shown that some techniques developed in robust statis-

Manuscript received March 12, 1992; revised December 28, 1992. This work was supported by DARPA and the U.S. Air Force under Contracts AFOSR-89-0506 and F4969092-J-0466.

L. Xu was on leave at the Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139. He is with the Department of Computer Science, The Chinese University of Hong Kong, Shatin, Hong Kong, and the Information Science Center, Peking University, Peking, People's Republic of China.

A. L. Yuille is with the Harvard Robotics Laboratory, Harvard University, Cambridge, MA 02138 USA.

IEEE Log Number 9207789.

[1] Except the algorithm given in [37], where outliers can be resisted to some extent.

tics (e.g., redescending M-estimators, least-trimmed squares estimators) appear naturally within the Bayesian formulation by the use of the statistical physics approach. In this paper, we adapt this approach to tackle the problem of robust PCA. First, we connect some existing PCA learning rules [22], [25], [36] to energy functions. Then we generalize the energy function by adding a binary decision field with a given prior distribution so that outliers are taken into consideration. Each generalized energy function is further used to define a Gibbs distribution from which a marginal distribution is obtained by summing over the binary decision field. This marginal distribution further defines an effective energy function, which is used to derive self-organizing rules for robust PCA. Computer experiments have been made comparing the obtained robust self-organizing rules with several existing PCA algorithms, including the standard eigenvector analyzing algorithm and those algorithms given in [22], [25], [36]. These experiments showed that in the presence of outliers, the eigenvector-analyzing algorithm performs very badly, the algorithms given in [22], [25], [36] work better but still give quite poor results, while the robust rules developed in this paper have improved the performances of the existing PCA algorithms significantly.

From Sections II to IV, we study the robust PCA problem of finding only the first principal component. In Section II, some existing PCA algorithms are introduced and then connected to energy functions. Section III shows how these energy functions are used to form the corresponding Gibbs distributions, and then to determine the related effective energy function by the statistical physics approach. Thereafter, robust self-organizing *batch way* rules for PCA are derived based on the resulting effective energy functions. Furthermore, in Section IV, the robust *batch way* rules are further converted into *on-line way* or *adaptive* rules. Results obtained for solving for the first principal component vector are given, in comparative experiments, to show the significant improvements achieved by the proposed robust rules. In Section V, we generalize the robust rules obtained in the previous sections to the cases of solving for the first $k(\geq 2)$ principal components or directly for the subspace spanned by the first k principal component vectors without solving for each of these vectors individually. Again, the results of comparative experiments show the success of the proposed robust rules for resisting outliers. Section VI gives further remarks on the selection of parameters used in the proposed rules, on making minor component analysis robust, and on other existing robust PCA algorithms developed in the literature of statistics.

II. PCA, Self-Organizing Rules, and Energy Functions

Assume that $\vec{x}$ is an n-dimensional random vector with zero mean $E\{\vec{x}\} = 0$. The dot product $\vec{\phi}^T\vec{x}$ is called the principal component if $E\{(\vec{\phi}^T\vec{x})^2\} = \text{Max}_{\vec{m}} E\{(\vec{m}^T\vec{x})^2\}$ with the constraint $\vec{m}^T\vec{m} = 1$. The solution for the vector $\vec{m}$ is the first dominant eigenvector $\vec{\phi}$ of the data covariance matrix, given by

$$\Sigma\vec{\phi} = \lambda\vec{\phi}, \qquad \text{with } \Sigma = E\{\vec{x}\vec{x}^T\} \tag{1}$$

where λ is the largest eigenvalue of Σ. Correspondingly, the process of finding $\vec{\phi}$ (or equivalently, λ) is called principal component analysis (PCA).

For a given data set $\{\vec{x}_i,\ i = 1, \cdots, N\}$ with zero mean $(1/N)\sum_{i=1}^{N}\vec{x}_i = 0$, a simple approach to do PCA is given as follows.

1) Compute the simple variance matrix

$$S = \frac{1}{N}\sum_{i=1}^{N}\vec{x}_i\vec{x}_i^T. \tag{2}$$

2) Use any of the existing standard eigenvector analyzing algorithms (e.g., Jacobs method) to solve

$$S\vec{\phi} = \lambda\vec{\phi} \tag{3}$$

where λ and $\vec{\phi}$ is the largest eigenvalue and its corresponding eigenvector, respectively.

This standard eigenvector-analyzing-based approach has two major problems. First, the solution of $S\vec{\phi} = \lambda\vec{\phi}$ is made after all the data have been collected and S has been calculated, i.e., the approach works in the *batch* way. When a new sample $\vec{x}'$ is added, although it is not difficult to get the corresponding new variance matrix by $S' = (NS + \vec{x}'\vec{x}'^T)/(N+1)$ with a small amount of computation, all the computations for solving $S\vec{\phi} = \lambda\vec{\phi}$ need to be repeated to solve $S'\vec{\phi} = \lambda\vec{\phi}$. So the approach is not suitable for real applications where data come incrementally or in the *on-line* way. Second, as will be shown by the results of experiments, the approach will deteriorate drastically and produce unacceptable results on a data set with outliers. This unfavorable feature has also been well shown in Huber's book [14]. He has demonstrated that even one outlier can by itself determine an entire component. Worse still, this component can even have one of the largest eigenvalues.

The first problem can be solved by a number of existing self-organizing rules for PCA [22], [24], [36], [39]. Three such rules are listed as follows:

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_a(t)(\vec{x}y - \vec{m}(t)y^2) \tag{4}$$

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_a(t)\left(\vec{x}y - \frac{\vec{m}(t)}{\vec{m}(t)^T\vec{m}(t)}y^2\right) \tag{5}$$

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_a(t)[y(\vec{x} - \vec{u}) + (y - y')\vec{x}] \tag{6}$$

where $y = \vec{m}(t)^T\vec{x}$, $\vec{u} = y\vec{m}(t)$, $y' = \vec{m}(t)^T\vec{u}$, and $\alpha_a(t) \geq 0$ is the learning rate which decreases to zero as $t \to \infty$ while satisfying certain conditions, e.g.,

$$\sum_t \alpha_a(t) = \infty, \quad \sum_t \alpha_a(t)^q < \infty \qquad \text{for some } q > 1. \tag{7}$$

Each of the three rules will converge to the principal component vector $\vec{\phi}$ almost surely under some mild conditions which are studied in detail in [22], [24], [36], [39]. By regarding $\vec{m}$ as the weight vector (i.e., the vector consisting of synapses) of a linear neuron with output $y = \vec{m}^T\vec{x}$, all of the three rules can be considered as modifications of the well-known Hebbian rule

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_a(t)\vec{x}y \tag{8}$$

for self-organizing the synapses of a neuron. In comparison with the original Hebbian rule, (4)–(6) each has an additional term for preventing $\|\vec{m}(t)\|$ from going to ∞ as $t \to \infty$.

Among the three rules, (4) requires the smallest number of computations for each updating of $\vec{m}(t)$. Moreover, the updating of each component $m^{(i)}(t)$ of $\vec{m}^{(t)}$ is just based on locally available variables $m^{(i)}(t)$ and y. This kind of locality is usually regarded as being "more biologically plausible" [13]. Equation (5) is slightly different from (4) in that an explicit normalization term $\vec{m}(t)^T\vec{m}(t)$ is used. This destroys the locality property of (4) since the updating of $m^{(i)}(t)$ also involves $m^{(j)}(t)$ for all $j \neq i$. However, we can show that (5) directly relates to an energy function which can provide a bridge for generalizing the rules (4), (5) into robust versions by using the statistical physics approach. Equation (6) is also local in the sense that the updating of $m^{(i)}(t)$ only involves $m^{(i)}(t)$, y, y', and $u^{(i)}$, all of which are locally available at the synapse corresponding to this $m^{(i)}(t)$. Moreover, (6) also possesses the advantage that it directly relates to an energy function, and thus can be generalized into a robust version by the statistical physics approach.

As will be shown later by the experimental results given in Section IV, the above self-organizing rules (4)–(6) can also partly solve the second problem of the simple approach given by (2) and (3), i.e., the problem of being sensitive to outliers. Although some better outlier-resisting versions of (4) and (5) have also been recently proposed in [37], they work well only for data which are not severely spoiled by outliers. Here, we adopt an approach which is totally different from that used in [37]—we generalize (4)–(6) into more robust versions by using the statistical physics approach.

In order to use the statistical physics approach, we need to connect these rules to energy functions. First, we show that (5) directly relates to the following energy function:

$$J_1(\vec{m}) = E\left\{\vec{x}^T\vec{x} - \frac{y^2}{\vec{m}^T\vec{m}}\right\} = tr\Sigma - \frac{\vec{m}^T\Sigma\vec{m}}{\vec{m}^T\vec{m}} \tag{9}$$

or, in terms of a given sample set $\vec{x}_i$, $i = 1, \cdots, N$,

$$J_1(\vec{m}) = \frac{1}{N}\sum_{i=1}^{N}\left(\vec{x}_i^T\vec{x}_i - \frac{\vec{m}^T\vec{x}_i\vec{x}_i^T\vec{m}}{\vec{m}^T\vec{m}}\right). \tag{10}$$

(Note: it is not difficult to see that $J_1(\vec{m}) \geq 0$ since $\vec{x}^T\vec{x} - (y^2/\vec{m}^T\vec{m}) = \|\vec{x}\|^2\sin^2\theta_{xm} \geq 0$, where $y = \vec{m}^T\vec{x}$ and θ_{xm} is the angle between $\vec{m}$, $\vec{x}$.)

On the one hand, the gradient descent rule for minimizing $J_1(\vec{m})$ is

$$\frac{d\vec{m}}{dt} = -\frac{\partial J_1(\vec{m})}{\partial\vec{m}} = \frac{2}{\vec{m}^T\vec{m}}\left(\Sigma\vec{m} - \frac{\vec{m}^T\Sigma\vec{m}}{\vec{m}^T\vec{m}}\vec{m}\right). \tag{11}$$

On the other hand, by taking expectation on (5) and noticing that $\vec{m}(t)$ changes much more slowly than $\vec{x}$, we have

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_a(t)\left(\Sigma\vec{m}(t) - \frac{\vec{m}(t)^T\Sigma\vec{m}(t)}{\vec{m}(t)^T\vec{m}(t)}\vec{m}(t)\right). \tag{12}$$

This is just the discrete form of (11) if the learning rate $\alpha_a(t)$ is appropriately selected. In other words, (5) is the *adaptive* rule (also called the *on-line* rule or the *stochastic approximation* rule) for minimizing $J_1(\vec{m})$ in the gradient descent manner.

Let us further show that (6) is an adaptive rule for minimizing the following energy in the gradient descent manner:

$$J_2(\vec{m}) = E(\|\vec{x} - \vec{u}\|^2)$$

or

$$J_2(\vec{m}) = \left(\frac{1}{N}\sum_{i=1}^{N}\|\vec{x}_i - \vec{u}_i\|^2\right) \tag{13}$$

where $\vec{u} = y\vec{m}$ can be regarded as a reconstructed pattern of input $\vec{x}$ from the output y, and minimizing $J_2(\vec{m})$ is equivalent to the problem of reconstructing $\vec{x}$ in the least square sense [36].

The gradient descent rule for minimizing $J_2(\vec{m})$ is given by

$$\begin{aligned}\frac{d\vec{m}}{dt} &= -\frac{\partial J_2(\vec{m})}{\partial\vec{m}} \\ &= -\frac{\partial E(\|\vec{x}\|^2 - 2\vec{m}^T\vec{x}\vec{x}^T\vec{m} + \vec{m}^T\vec{m}\vec{m}^T\vec{x}\vec{x}^T\vec{m})}{\partial\vec{m}} \\ &= -\frac{\partial(tr\Sigma - 2\vec{m}^T\Sigma\vec{m} + \vec{m}^T\vec{m}\vec{m}^T\Sigma\vec{m})}{\partial\vec{m}} \\ &= 2[(1 - \|\vec{m}\|^2)\Sigma\vec{m} + \Sigma\vec{m} - (\vec{m}^T\Sigma\vec{m})\vec{m}].\end{aligned} \tag{14}$$

On the other hand, by taking the expectation of (6), we get

$$\begin{aligned}&\vec{m}(t+1) \\ &= \vec{m}(t) + \alpha_a(t)E[y(\vec{x} - \vec{u}) + (y - y')\vec{x}] \\ &= \vec{m}(t) + \alpha_a(t)[\Sigma\vec{m}(t) - (\vec{m}(t)^T\Sigma\vec{m}(t))\vec{m}(t) \\ &\quad + \Sigma\vec{m}(t) - (\vec{m}(t)^T\vec{m}(t))\Sigma\vec{m}(t)].\end{aligned} \tag{15}$$

Again, we see that (15) is the discrete form of (14). That is, (6) is an adaptive rule for minimizing $J_2(\vec{m})$ in the gradient descent manner.

Finally, let us consider (4). As pointed out in [2], the rule given by (4) is not a gradient descent rule of any kind of energy function. However, in [36], [39], one of the present authors has recently proved the following results.

1) Let $\vec{h}_1 = \vec{x}y - \vec{m}y^2$, $\vec{h}_2 = \vec{x}y - (\vec{m}/\vec{m}^T\vec{m})y^2$; then $\vec{h}_1^T\vec{h}_2 \geq 0$, $E(\vec{h}_1)^TE(\vec{h}_1) \geq 0$.

2) Let $\vec{h}_3 = y(\vec{x} - \vec{u}) + (y - y')\vec{x}$; then $E(\vec{h}_1)^TE(\vec{h}_3) \geq 0$.

3) Both J_1 and J_2 have only one local (also global) minimum $\text{tr}(\Sigma) - \vec{\phi}^T\Sigma\vec{\phi}$, and all the other critical points (i.e., the points that satisfy $\partial J_i(\vec{m})/\partial\vec{m} = 0$, $i = 1, 2$) are saddle points.

The above results revealed that (4) is a downhill algorithm for minimizing J_1 in both the *on-line* sense and *average* sense, and for minimizing J_2 in the average sense. Since the principal component vector $\vec{\phi}$ is the only local minimum of J_i, $i = 1, 2$, the adaptive rules (4)–(6) will finally reach the same solution. Therefore, we can also connect the rule (4) to either J_1 or J_2.

III. Robust PCA by Techniques of Statistical Physics

For a data set $\{\vec{x}_i, i = 1, \cdots, N\}$, the energy functions $J_1(\vec{m})$, $J_2(\vec{m})$ given by (10) and (13) can be regarded as

special cases of the following general energy function:

$$J(\vec{m}) = \frac{1}{N}\sum_{i=1}^{N} z(\vec{x}_i, \vec{m}), \qquad z(\vec{x}_i, \vec{m}) \geq 0 \tag{16}$$

where $z(\vec{x}_i, \vec{m})$ is the portion of energy contributed by the sample $\vec{x}_i$, and

$$\begin{aligned} z(\vec{x}_i, \vec{m}) &= \left(\vec{x}_i^T \vec{x}_i - \frac{\vec{m}^T \vec{x}_i \vec{x}_i^T \vec{m}}{\vec{m}^T \vec{m}}\right) \qquad \text{for } J_1 \\ &= \|\vec{x}_i - \vec{u}_i\|^2 \qquad \text{for } J_2. \end{aligned} \tag{17}$$

Following [42], [43], we now generalize energy (16) into

$$E(\vec{V}, \vec{m}) = \sum_{i=1}^{N} V_i z(\vec{x}_i, \vec{m}) + E_{\text{prior}}(\vec{V}) \tag{18}$$

where $\vec{V} = \{V_i,\ i = 1, \cdots, N\}$ is a binary field $\{V_i\}$, with each V_i being a random variable taking value either 0 or 1. V_i acts as a decision indicator for deciding whether $\vec{x}_i$ is an outlier or a sample. When $V_i = 1$, the portion of energy contributed by the sample $\vec{x}_i$ is taken into consideration; otherwise, it is equivalent to discarding $\vec{x}_i$ as an outlier. $E_{\text{prior}}(\vec{V})$ is the *a priori* portion of energy contributed by the prior distribution of $\{V_i\}$. A natural choice is

$$E_{\text{prior}}(\vec{V}) = \eta\sum_{i=1}^{N}(1 - V_i). \tag{19}$$

This choice of prior has a natural interpretation: for fixed $\vec{m}$, it is energetically favorable to set $V_i = 1$ (i.e., not to regard $\vec{x}_i$ as an outlier) if $z(\vec{x}_i, \vec{m}) < \sqrt{\eta}$ (i.e., the portion of energy contributed by $\vec{x}_i$ is smaller than a prespecified threshold) and to set it to 0 otherwise.

The goal is to minimize $E[\vec{V}, \vec{m}]$ with respect to $\{V_i\}$ and $\vec{m}$ simultaneously while obeying the constraint that every V_i only takes binary value. This problem is a mixture of discrete and continuous optimization. The solution usually cannot be calculated analytically, and is also hard to obtain by the gradient descent approach. Although, by setting $\partial E(\vec{V}, \vec{m})/\partial \vec{m} = 0$ and solving for the $\vec{m}$ in terms of the $\{V_i\}$, one may transform the problem into minimizing a function of the $\{V_i\}$ subject to some constraints, it still leads to a hard discrete optimization problem.

To help us solve this problem, we define a Gibbs distribution [30]:

$$P[\vec{V}, \vec{m}] = \frac{1}{Z} e^{-\beta E[\vec{V}, \vec{m}]} \tag{20}$$

where Z is the partition function which ensures $\sum_{\vec{V}} \int_{\vec{m}} P[\vec{V}, \vec{m}] = 1$. Now, minimizing $E[\vec{V}, \vec{m}]$ is equivalent to maximizing $P[\vec{V}, \vec{m}]$. But this still does not eliminate the difficulty of discrete optimization with respect to the binary variables $\{V_i\}$. One solution for the problem is to compute the marginal probability distribution $P_{\text{margin}}(\vec{m})$ by averaging out these variables $\{V_i\}$, taking into consideration the constraint that they only take binary values, and then to use the maximization of $P_{\text{margin}}(\vec{m})$ to approximate the maximization of $P(\vec{V}, \vec{m})$. This is equivalent to computing the mean field approximation to the statistical physics system by the saddle point method [30]. Analytically, $P_{\text{margin}}(\vec{m})$ can be computed as follows:

$$\begin{aligned} P_{\text{margin}}(\vec{m}) &= \frac{1}{Z}\sum_{\vec{V}} e^{-\beta \sum_i \{V_i z(\vec{x}_i, \vec{m}) + \eta(1 - V_i)\}} \\ &= \frac{1}{Z}\prod_i \sum_{V_i = \{0, 1\}} e^{-\beta\{V_i z(\vec{x}_i, \vec{m}) + \eta(1 - V_i)\}} \\ &= \frac{1}{Z}\prod_i \{e^{-\beta\eta} + e^{-\beta z(\vec{x}_i, \vec{m})}\} \\ &= \frac{e^{-N\beta\eta}}{Z}\prod_i \{1 + e^{-\beta\{z(\vec{x}_i, \vec{m}) - \eta\}}\}. \end{aligned} \tag{21}$$

Defining $Z_m = Ze^{N\beta\eta}$, we obtain

$$P_{\text{margin}}(\vec{m}) = \frac{1}{Z_m} e^{-\beta E_{\text{eff}}(\vec{m})} \tag{22}$$

where

$$E_{\text{eff}}(\vec{m}) = \frac{-1}{\beta}\sum_i \log\{1 + e^{-\beta\{z(\vec{x}_i, \vec{m}) - \eta\}}\}. \tag{23}$$

Maximizing $P_{\text{margin}}(\vec{m})$ with respect to x is equivalent to minimizing $E_{\text{eff}}(\vec{m})$. The form of E_{eff} can be regarded as a generalization of a robust redescending M-estimators [14] to the PCA problem. It is clear that each term in the sum for E_{eff} is just $z(\vec{x}_i, \vec{m})$ if it has a small value, but becomes constant as $z(\vec{x}_i, \vec{m}) \to \infty$. In this way, outliers which are more likely to yield large $z(\vec{x}_i, \vec{m})$ are treated differently from samples, and thus the estimation $\vec{m}$ obtained by minimizing $E_{\text{eff}}(\vec{m})$ will be robust and able to resist outliers.

$E_{\text{eff}}(\vec{m})$ is usually not a convex function, and may have many local minima. The statistical physics framework suggests using deterministic annealing to minimize $E_{\text{eff}}(\vec{m})$. That is, by the following gradient descent rule (24), to minimize $E_{\text{eff}}(\vec{m})$ for small β and then track the minimum as β increases to infinity (the zero temperature limit):

$$\begin{aligned} \frac{\partial \vec{m}}{\partial t} &= -\frac{\partial E_{\text{eff}}(\vec{m})}{\partial \vec{m}} \\ \vec{m}(t+1) &= \vec{m}(t) - \alpha_b(t)\frac{\partial E_{\text{eff}}(\vec{m}(t))}{\partial \vec{m}(t)} \\ &= \vec{m}(t) - \alpha_b(t)\sum_i \frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \\ &\quad \cdot \frac{\partial z(\vec{x}_i, \vec{m}(t))}{\partial \vec{m}(t)} \end{aligned} \tag{24}$$

where $\alpha_b(t)$ is the learning rate, again it satisfies some condition like (7).

Specifically, corresponding to the energies J_1, J_2 given in (9) and (13), we have the following *batch* way for PCA:

$$\begin{aligned} \vec{m}(t+1) &= \vec{m}(t) + \alpha_b(t)\sum_i \frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \\ &\quad \cdot \left(\vec{x}_i y_i - \frac{\vec{m}(t)}{\vec{m}(t)^T \vec{m}(t)} y_i^2\right) \end{aligned} \tag{25}$$

$$\begin{aligned} \vec{m}(t+1) &= \vec{m}(t) + \alpha_b(t)\sum_i \frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \\ &\quad \cdot [y_i(\vec{x}_i - \vec{u}_i) + (y_i - y_i')\vec{x}_i] \end{aligned} \tag{26}$$

where, as $t \to \infty$, the learning rate $\alpha_b(t) \to 0$ and the annealing parameter $\beta \to \infty$.

Finally, the converged vector $\vec{m}_{\text{converge}}$ is taken as the resulted principal component vector which has avoided the effects of outliers. In addition, a byproduct can be easily obtained by

$$\begin{aligned} V_i &= 1, \quad \text{if } z(\vec{x}_i, \vec{m}_{\text{converge}}) < \sqrt{\eta} \\ &= 0, \quad \text{otherwise} \end{aligned} \tag{27}$$

which indicates whether $\vec{x}_i$ is an outlier ($V_i = 0$) or not ($V_i = 0$).

Before closing this section, we would also like to make a discussion on the relationship of our approach to maximal likelihood (ML) estimation of finite mixture distributions.

Note that we can also rewrite (23) in the following form:

$$\begin{aligned} P_{\text{margin}}(\vec{m}) &= \frac{1}{Z}\prod_i \{e^{-\beta\eta} + e^{-\beta z(\vec{x}_i, \vec{m})}\} = \prod_i f(\vec{x}_i, \vec{m}) \\ &= \prod_i \{a f_1(\vec{x}_i, \vec{m}) + b f_2(\vec{x}_i, \vec{m})\} \end{aligned} \tag{28}$$

with a, b being constants which depend only on β, η, but are independent of $\vec{x}_i$, $\vec{m}$, and

$$f_1(\vec{x}_i, \vec{m}) = C_1 \frac{1}{Z}, \qquad f_2(\vec{x}_i, \vec{m}) = C_2 \frac{e^{-\beta z(\vec{x}_i, \vec{m})}}{Z}$$

where C_1, C_2 are normalizing constants which ensure that $\int f_1(\vec{x}_i, \vec{m})\, d\vec{x}_i = 1$, $\int f_2(\vec{x}_i, \vec{m})\, d\vec{x}_i = 1$. Because $f_1(\vec{x}_i, \vec{m})$ is constant, we must require that the integral is taken over a compact domain enclosing the datapoints or, equivalently, redefine $f_1(\vec{x}_i, \vec{m})$ so that it vanishes outside this compact domain.

It follows from (28) that maximizing $P_{\text{margin}}(\vec{m})$ with respect to $\vec{m}$ is equivalent to solving a maximum-likelihood (ML) estimation of a mixture distribution [31], [15] of $f_1(\vec{x}_i, \vec{m})$ and $f_2(\vec{x}_i, \vec{m})$. Clearly, $f_1(\vec{x}_i, \vec{m})$ is a uniform distribution and $f_2(\vec{x}_i, \vec{m})$ is a distribution that depends on our choice of $z(\vec{x}_i, \vec{m})$. Thus, from the ML perspective, our model assumes that the outliers are generated by a uniform distribution and the data are generated by a process that depends on our specific choice of $z(\vec{x}_i, \vec{m})$. Note that this is a Gaussian for both of the choices described by (17).

IV. Robust Adaptive Rules and Computer Experiments

The rules (36), (25), (26), like the simple approach given by (2), (3), work in the *batch* way, and thus are not suitable for situations where data come incrementally, in the *on-line* way. Now, let us convert them into adaptive versions and, at the same time, maintain their robustness in the presence of outliers.

One way to reach this goal is quite simple: just remove the summation in (24), or equivalently, minimize the energy portion contributed by the current sample $\vec{x}_i$:

$$e(\vec{m}, \vec{x}_i) = \frac{-1}{\beta}\log\{1 + e^{-\beta\{z(\vec{x}_i, \vec{m}) - \eta\}}\}. \tag{29}$$

By the gradient descent approach, we can get the following general adaptive rule:

$$\vec{m}(t+1) = \vec{m}(t) - \alpha_a(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}}\frac{\partial z(\vec{x}_i, \vec{m}(t))}{\partial \vec{m}(t)}. \tag{30}$$

By taking the expectation of both sides of (24) and (30), under the assumption that $\vec{x}_i$ comes from a stationary process and that $\vec{m}(t)$ changes much slower than $\vec{x}_i$, we can obtain the equation

$$\begin{aligned} &\vec{m}(t+1) \\ &= \vec{m}(t) - \alpha(t) E\left\{\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}}\frac{\partial z(\vec{x}_i, \vec{m}(t))}{\partial \vec{m}(t)}\right\} \end{aligned} \tag{31}$$

where $\alpha(t) = \alpha_a(t) = N\alpha_b(t)$. This means that (30) is an adaptive or stochastic approximation rule which minimizes $E_{\text{eff}}(\vec{m})$ of (23) in the gradient descent manner.

According to the specific forms of $z(\vec{x}_i, \vec{m})$ for J_1, J_2, we can rewrite (30) into the following *adaptive* self-organizing rules for PCA:

$$\begin{aligned} \vec{m}(t+1) = \vec{m}(t) &+ \alpha_a(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \\ &\cdot \left(\vec{x}_i y_i - \frac{\vec{m}(t)}{\vec{m}(t)^T\vec{m}(t)} y_i^2\right) \end{aligned} \tag{32}$$

$$\begin{aligned} \vec{m}(t+1) = \vec{m}(t) &+ \alpha_a(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \\ &\cdot [y_i(\vec{x}_i - \vec{u}_i) + (y_i - y_i')\vec{x}_i]. \end{aligned} \tag{33}$$

We can observe that the difference between (5) and (32) or (6) and (33) is that the learning rate $\alpha_a(t)$ has been modified by a multiplicative factor

$$\alpha_m(t) = \frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}} \tag{34}$$

which adaptively modifies the learning rate to suit the current input $\vec{x}_i$. This modifying factor has a similar function to that used by one of the present authors for robust line fitting [37]. But the modifying factor (34) is more sophisticated and performs better.

Based on the connection between (4) and energy function J_1 or J_2 (discussed at the end of Section II), we can formally use the modifying factor $\alpha_m(t)$ to turn the rule (4) into the following robust version:

$$\begin{aligned} &\vec{m}(t+1) \\ &= \vec{m}(t) + \alpha_a(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}}(\vec{x}_i y_i - \vec{m}(t) y_i^2), \end{aligned} \tag{35}$$

and the corresponding *batch way* rule is given as follows:

$$\begin{aligned} \vec{m}(t+1) = \vec{m}(t) &\\ + \alpha_b(t)\sum_i &\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}(t)) - \eta)}}(\vec{x}_i y_i - \vec{m}(t) y_i^2) \end{aligned} \tag{36}$$

where $z(\vec{x}_i, \vec{m}(t)) = (\vec{x}_i^T\vec{x}_i - (\vec{m}^T(t)\vec{x}_i\vec{x}_i^T\vec{m}(t)/\vec{m}^T(t)\vec{m}(t))$.

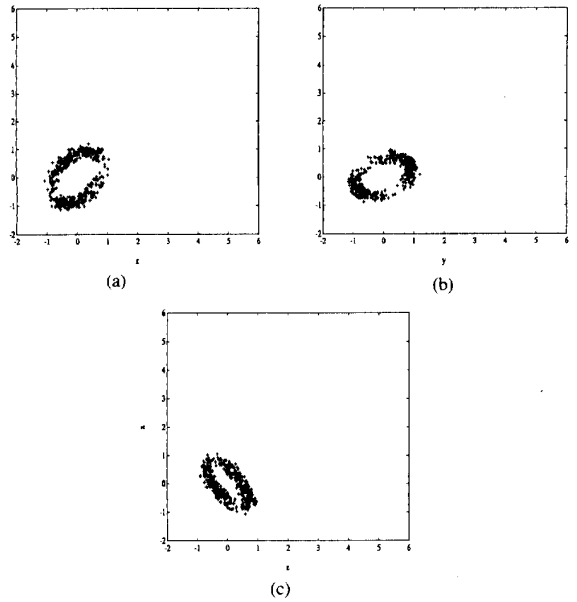


Fig. 1. The projections of data samples on the x–y, y–z, and z–x planes, without outliers.

In the rest of this section, we introduce some results obtained from comparative experiments on the rules given in Section II and their robust versions given earlier in this section.

Let $\vec{x}$ be a 3-D vector coming from a 3-D population of 400 samples with zero mean. These samples are located on an elliptic ring which is centered at the origin of the R^3 space, with the largest elliptic axis being along the direction $(-1, 1, 0)$, the plane of the two elliptic axes intersecting the x–y plane with an acute angle (30°). The projections of the 400 samples on the x–y, y–z, and z–x planes are shown in Fig. 1(a)–(c), respectively.

Without the presence of outliers, the simple approach given by (2), (3) finds that the principal component vector of this data set is $\vec{\phi}_p = [0.0710, 0.8876, 0.4551]^T$. The results of using (4)–(6) are given in Fig. 2, where $UA1$, $UA2$, $UA3$ denote the unrobust adaptive rules (4), (5), and (6), respectively, and the vertical axis denotes the angle θ between the present $m(t)$ and $\vec{\phi}_p$:

$$\theta = \theta', \text{ where } 0 \le \theta \le 90°; \quad \theta = 180° - \theta', \text{ otherwise;}$$
$$\text{and } \theta' = \cos^{-1}\left(\frac{\vec{\phi}_p^T \vec{m}(t)}{\|\vec{\phi}_p\| \, \|\vec{m}(t)\|}\right). \quad (37)$$

We see that all three rules converge to $\vec{\phi}_p$ quite perfectly.

Next, the data set is contaminated by ten outlier points (the amount of contamination is only 2.5% of the data set). The projections of this spoiled data set on the x–y, y–z, and z–x planes are shown in Fig. 3(a)–(c). This time, the result of the simple approach given by (2), (3) is $\vec{\phi} = [-0.8285, 0.1567, 0.5375]^T$ and the angle between $\vec{\phi}$ and the right one $\vec{\phi}_p$ is 71.04°. It is obvious that the result is absolutely unacceptable. This reveals that this simple approach has no outlier-resisting ability.

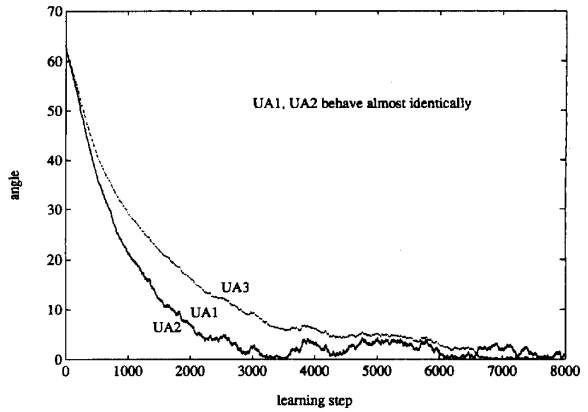


Fig. 2. The learning curves in the comparative experiments for the first principal component vector using the unrobust adaptive rules, (4), (5), and (6), denoted by $UA1$, $UA2$, $UA3$, respectively.

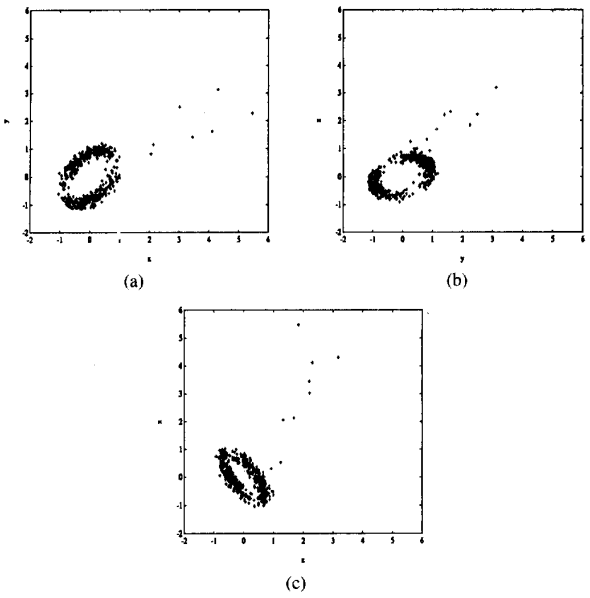


Fig. 3. The projections of data samples on the x–y, y–z, and z–x planes, with ten outlier points.

Fig. 4 gives the results of using the robust *batch way* rules (35), (25), (26), and their unrobust counterparts:

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_b(t)\sum_i(\vec{x}_i y_i - \vec{m}(t)y_i^2) \quad (38)$$

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_b(t)\sum_i\left(\vec{x}_i y_i - \frac{\vec{m}_t}{\vec{m}(t)^T\vec{m}(t)}y_i^2\right) \quad (39)$$

$$\vec{m}(t+1) = \vec{m}(t) + \alpha_b(t)\sum_i[y_i(\vec{x}_i - \vec{u}_i) + (y_i - y_i')\vec{x}_i]. \quad (40)$$

In Fig. 4, $RB1$, $RB2$, $RB3$ denote the *robust batch way* rules (36), (25), and (26), respectively, and $UB1$, $UB2$, $UB3$ denote the *unrobust batch way* rules (38), (39), and (40),

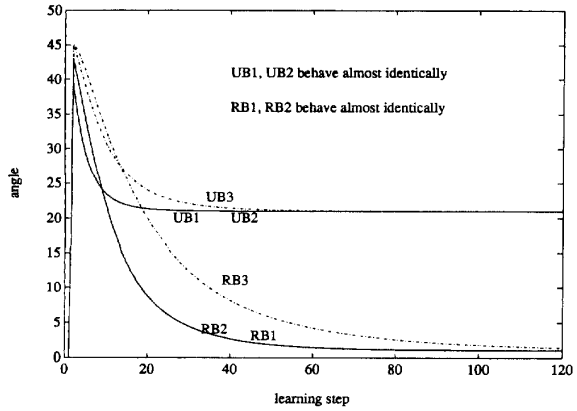


Fig. 4. The learning curves obtained in the comparative experiments for the first principal component vector by robust and unrobust rules working in the *Batch* way. $RB1$, $RB2$, $RB3$ denote the *robust batch* rules, (35), (25), and (26), respectively, and $UB1$, $UB2$, $UB3$ denote the *unrobust batch* rules, (38), (39), and (40), respectively.

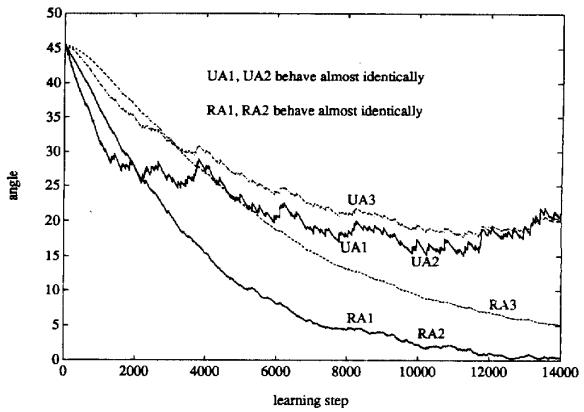


Fig. 5. The learning curves obtained in the comparative experiments for the first principal component vector by robust and unrobust rules working in the *adaptive* way. $RA1$, $RA2$, $RA3$ denote the *robust adaptive* rules, (35), (32), and (33), respectively. As in Fig. 2, $UA1$, $UA2$, $UA3$ denote the *unrobust adaptive* rules, (4), (5), and (6), respectively.

respectively. It can be seen that all of the three rules converge to a vector which has an angle of 20.97° to the correct $\vec{\phi}_p$. The result is certainly much better than that obtained by the simple approach given by (2), (3). This means that the rules $UB1$, $UB2$, $UB3$ do have some outlier-resisting ability. However, this solution still has a big error. By contrast, the learning process by all the robust rules $RB1$, $RB2$, $RB3$ converges to a vector which has an angle of only about 1.06° to the correct $\vec{\phi}_p$. The solution is very accurate and significantly better than that obtained by the unrobust rules $UB1$, $UB2$, $UB3$. This shows that the robust rules $RB1$, $RB2$, $RB3$ do resist outliers very well. In this experiment, the parameters used are $\alpha_b(t) = 0.3$, $\beta = 0.5$, and $\eta = 4$ for all the rules. Here, for simplicity, we kept these parameters constant. We would expect that better solutions could be obtained when $\alpha \to 0$, $\beta \to \infty$, $\eta \to 0$ in a suitable way.

The results of using the robust *adaptive* rules (35), (32), and (33) and their unrobust counterparts—the rules (4), (5), and (6)—are shown in Fig. 5, in which $RA1$, $RA2$, $RA3$ denote the *robust adaptive* rules (35), (32), and (33), respectively, and like in Fig. 2, $UA1$, $UA2$, $UA3$ denote the *unrobust adaptive* rules (4), (5), and (6), respectively.

We again see that all three unrobust rules converge to a vector which has an angle of about 21° to the correct $\vec{\phi}_p$, while the learning process by the robust rules $RA1$, $RA2$, $RA3$ converge to a very accurate solution which has an angle of only about 0.36° to the correct $\vec{\phi}_p$. This again revealed that the robust rules proposed in this paper have improved the outlier-resisting ability of the conventional PCA algorithms significantly. In this example, the parameters used are $\alpha_a(t) = 0.001$, $\beta = 0.5$, and $\eta = 4$, respectively.

From Figs. 2, 4, and 5, we can also observe that the normalized versions $UA2$, $UB2$, $RA2$, $RB2$ behave almost identically to their unnormalized counterparts $UA1$, $UB1$, $RA1$, $RB1$. This suggests that the unnormalized versions are better choices than the normalized ones because the unnormalized versions can save the computation of $m(t)^T m(t)$ at each learning step. In addition, Figs. 2, 4, and 5 also show that either $UA1$, $UB1$, $RA1$, $RB1$ or $UA2$, $UB2$, $RA2$, $RB2$ converge faster but fluctuate more than $UA3$, $UB3$, $RA3$, $RB3$. Does this means that the third set of rules is inferior to the first two sets of rules in the sense of converging speed, but is superior to the two sets in the sense of low variations? The answer is negative. Our further experiments have shown that the convergence speed and fluctuation of every set of rules depends on the value of the learning rate. The larger the rate, the faster the convergence and the bigger the fluctuation. The scale of the rate to each set of rules is different. Even more interestingly, the three sets of rules can behave almost identically by appropriately scaling the learning rate used in each set. For example, when we let the learning rate used by the third set be two times as large as that used by the first two sets in an experiment similar to that shown in Fig. 4, we find that the learning processes of all of the three sets are almost the same. Therefore, we should consider that the three sets of rules are equally good in the sense of convergence speed. However, in the sense of computations consumed, the first set $UA1$, $UB1$, $RA1$, $RB1$ is the best. Thus, we recommend that this set be used in practice. Nevertheless, the other two sets are also of theoretical importance because of their roles for developing the robust versions in the previous sections.

V. Robust Analysis of k Principal Components and k-Dimensional Principal Subspace

In real applications, such as image processing and data communication, the first principal component is not sufficient and the first k principal components are used. They are defined as linear combinations $\vec{\phi}_j^T \vec{x}$, $j = 1, \cdots, k$ of the elements of $\vec{x}$ so that $E\{(\vec{\phi}_j^T \vec{x})^2\}$, $j = 1, \cdots, k$ are maximized under the constraints $\vec{\phi}_i^T \vec{\phi}_j = \delta_{ij}$ for $j < i$.

The solution for the vectors $\vec{\phi}_j$, $j = 1, \cdots, k$ are the k dominant eigenvectors of the data covariance matrix Σ given by (1). They are the k orthogonal unit vectors solved by

$\Sigma\vec{\phi}_j = \lambda_j\vec{\phi}_j$, $j = 1, \cdots, k$ with λ_j, $j = 1, \cdots, k$ being the k largest eigenvalues of Σ in descending order of magnitude. Correspondingly, the process of finding $\vec{\phi}_j$, $j = 1, \cdots, k$ (or equivalently, λ_j, $j = 1, \cdots, k$) is called k-principal component analysis (k-PCA).

In some applications, such as subspace pattern recognition [23], we do not need to get $\vec{\phi}_j$, $j = 1, \cdots, k$ individually, but only need to get the subspace spanned by these principal component vectors. We call such a subspace the k-dimensional principal subspace, and the process of obtaining this subspace the principal subspace analysis.

The section consists of two parts, each of which discusses how to generalize the robust rules developed in Section IV to implement the robust k-PCA, the k-dimensional principal subspace analysis, respectively.

Let us consider the first part. Now, we need k linear neurons with k weight vectors $\vec{m}_j$, $j = 1, \cdots, k$, and we expect that each $\vec{m}_j$ can converge to the jth principal component vector $\vec{\phi}_j$ during the learning process. By just directly using any of those rules developed in the previous sections on $\vec{m}_j$, $j = 1, \cdots, k$, we can expect that all of these vectors will converge to the same vector—the first principal component vector $\vec{\phi}_1$. This is not what we want; instead, we want that $\vec{m}_j$, $j = 1, \cdots, k$ should become orthogonal to each other and converge to k different principal component vectors $\vec{\phi}_j$, $j = 1, \cdots, k$.

There are two ways to reach this goal. One is the Gram–Schmidt orthonormalization (GSO)-based approach, like that used in the unrobust PCA algorithm SGA [27] and GHA [34]. The basic idea is to still use any one of the robust rules developed in the previous sections on $\vec{m}_j$, $j = 1, \cdots, k$, but at the same time also to use some orthonormalization procedure on these vectors after each learning step. Based on this basic idea, we can generalize the robust rules (35), (32), and (33) into the following general robust adaptive rule for k-PCA:

$$\vec{m}_j(t+1) = \vec{m}_j(t) + \alpha_a(t)\frac{1}{1+e^{\beta(z(\vec{x}_i(j),\vec{m}_j(t))-\eta)}}\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t)) \quad (41)$$

$$\vec{x}_i(0) = \vec{x}_i, \quad \vec{x}_i(j+1) = \vec{x}_i(j) - \sum_{r=1}^{j-1} y_i(r)\vec{m}_r(t) \quad (42)$$

$$y_i(j) = \vec{m}_j^T(t)\vec{x}_i(j) \quad (43)$$

where $\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t))$, $z(\vec{x}_i(j), \vec{m}_j(t))$ have the following four possibilities:

$$\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t)) = (\vec{x}_i(j)y_i(j) - \vec{m}_j(t)y_i(j)^2),$$

$$z(\vec{x}_i(j), \vec{m}_j(t)) = \vec{x}_i(j)^T\vec{x}_i(j) - \frac{y_i(j)^2}{\vec{m}_j(t)^T\vec{m}_j(t)}; \quad (44)$$

$$\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t)) = (\vec{x}_i(j)y_i(j) - \vec{m}_j(t)y_i(j)^2),$$

$$z(\vec{x}_i(j), \vec{m}_j(t)) = \|\vec{x}_i(j) - \vec{u}_i(j)\|^2; \quad (45)$$

$$\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t)) = \left(\vec{x}_i(j)y_i(j) - \frac{\vec{m}_j(t)}{\vec{m}_j(t)^T\vec{m}_j(t)}y_i(j)^2\right),$$

$$z(\vec{x}_i(j), \vec{m}_j(t)) = \vec{x}_i(j)^T\vec{x}_i(j) - \frac{y_i(j)^2}{\vec{m}_j(t)^T\vec{m}_j(t)}; \quad (46)$$

$$\Delta\vec{m}_j(\vec{x}_i(j), \vec{m}_j(t)) = [y_i(j)(\vec{x}_i(j) - \vec{u}_i(j)) + (y_i(j) - y_i(j)')\vec{x}_i(j)],$$

$$z(\vec{x}_i(j), \vec{m}_j(t)) = \|\vec{x}_i(j) - \vec{u}_i(j)\|^2 \quad (47)$$

where $\vec{u}_i(j) = y_i(j)\vec{m}_j(t)$, $y_i'(j) = \vec{m}_j^T(t)\vec{u}_i(j)$. Specially, for the combination (44) or (45), the general rule (41) can be regarded as the generalization of the unrobust k-PCA rule GHA [34].

The second way is to make the $\vec{m}_j$, $j = 1, \cdots, k$ become orthogonal to each other is the so-called asymmetric lateral anti-Hebbian learning method, like that used by Rubner [32]. Following [32], here we use an additional set of lateral weights w_{rj}, $r < j$ to laterally connect the rth neuron to the jth neuron such that the output of the jth neuron is modified by

$$y_i(j) = \vec{x}_i^t\vec{m}_j(t) - \sum_{r=1}^{j-1} w_{rj}y_i(r). \quad (48)$$

Again, any one of the robust rules developed in the previous sections can be used to modify $\vec{m}_j$, $j = 1, \cdots, k$, but now with $y_i(j)$ given by (48) instead of $y_i(j) = \vec{x}_i^T\vec{m}_j(t)$. At the same time, the classical anti-Hebbian learning rule is also used to modify the lateral weights w_{rj}, i.e.,

$$w_{rj}(t+1) = w_{rj}(t) - \alpha_a' y_i(r)y_i(j) \quad (49)$$

where α_a' is the learning rate. The rule will gradually decorrelate the output $y_i(j)$, $j = 1, \cdots, k$. After learning reaches its equilibrium, all the weights w_{rj} will vanish, and $\vec{m}_j$, $j = 1, \cdots, k$ will finally converge to the principal component vectors $\vec{\phi}_j$, $j = 1, \cdots, k$.

Fig. 6 is the results obtained by using the robust rule (41) with each of the possibilities (44) in comparison with its unrobust counterpart GHA [34] for solving for the first two principal component vectors. The data set is again the one used in Figs. 4 and 5 with outliers. The correct first two principal component vectors (i.e., the ones obtained on the data set without outliers) are $\vec{\phi}_{p1} = [0.0710, 0.8876, 0.4551]^T$ (which is the same as that obtained earlier in Section IV) and $\vec{\phi}_{p2} = [-0.7999, -0.2219, 0.5576]^T$. The solutions obtained by the unrobust simple rule (2) and (3) are $\vec{\phi}_1 = [-0.8285, 0.1567, 0.5375]^T$ and $\vec{\phi}_2 = [-0.3666, 0.5738 - 0.7324]^T$ with angles $\theta_1 = \text{angle}(\vec{\phi}_1, \vec{\phi}_{p1}) = 71.04°$, $\theta_2 = \text{angle}(\vec{\phi}_2, \vec{\phi}_{p2}) = 76.0°$, where $\theta_i = \text{angle}(\vec{\phi}_i, \vec{\phi}_{pi})$ denotes the angle between $\vec{\phi}_i$ and $\vec{\phi}_{pi}$, and is calculated in the same way as that in (37), respectively. Obviously, this result is absolutely unacceptable. In Fig. 6, $UAk1$, $UAk2$ denote the learning curves of the angles of $\theta_1 = \text{angle}(\vec{\phi}_1, \vec{\phi}_{p1})$ and $\theta_2 = \text{angle}(\vec{\phi}_2, \vec{\phi}_{p2})$, respectively, obtained by using the unrobust rule GHA [34]. *RAk1*, *RAk2* denote the learning curves of the angles obtained by using the robust rule (41).

It can be observed that both $UAk1$, $UAk2$ tend to fluctuate around an angle of approximately 23°. The result is certainly much better than that obtained by the simple standard approach (2) and (3), but still very poor. This reveals that the GHA [34] algorithm cannot resist outliers effectively. However, by our robust rule (41), $RAk1$, $RAk2$ converge to an angle of only about 1.7°, which is obviously a significant improvement from that obtained by GHA. In this example, the parameters used are $\alpha_a(t) = 0.003$, $\beta = 0.5$, and $\eta = 0.4$, respectively.

Now, let us proceed to the second part—the robust k-dimensional principal subspace analysis. Obviously, the job can be done by simply using the k principal component vectors $\vec{\phi}_j$, $j = 1, \cdots, k$, obtained by the above discussed methods, as the basis vectors to span a k-dimensional subspace. However, there are also some more direct ways, which can speed up convergence and save computation.

Let us denote $M = [\vec{m}_1, \cdots, \vec{m}_k]$, $\Phi = [\vec{\phi}_1, \cdots, \vec{\phi}_k]$, $\vec{y} = [y_1, \cdots, y_k]^T$, and $\vec{y} = M^T\vec{x}$. Then the unrobust rule (4) can be generalized into [25]

$$\vec{M}(t+1) = \vec{M}(t) + \alpha_A(t)(\vec{y}\vec{x}^T - \vec{y}\vec{y}^T M(t)) \tag{50}$$

and the unrobust rule (6) can be generalized into [36]

$$\vec{M}(t+1) = \vec{M}(t) + \alpha_A(t)(\vec{y}(\vec{x} - \vec{u})^T - (\vec{y} - \vec{y})\vec{x}^T) \tag{51}$$

where $\vec{u} = M\vec{y}$ and $\vec{y}' = M^T\vec{u}$.

In the case without outliers, by both the rules (50) and (51), the weight matrix $M(t)$ will converge to a matrix M^∞ whose column vectors m_j^∞, $j = 1, \cdots, k$ span the k-dimensional principal subspace [25], [36], although they are not equal to the k principal component vectors $\vec{\phi}_j$, $j = 1, \cdots, k$. In particular, they are even not orthogonal to each other.

As shown in [36], the rule (51) can be directly connected to the following energy function:

$$J_3(M) = E(\|\vec{x} - \vec{u}\|^2) \quad \text{or} \quad J_2(\vec{m}) = \left(\frac{1}{N}\sum_{i=1}^{N}\|\vec{x}_i - \vec{u}_i\|^2\right),$$

$$\vec{u} = M\vec{y}, \qquad \vec{y}' = M^T\vec{u}. \tag{52}$$

Actually, by taking the derivative $\partial E(\|\vec{x}-\vec{u}\|^2)/\partial M$ on one hand and taking the expectation on (51) on the other hand, it is not difficult to see that (51) is an *adaptive* or *stochastic approximation* rule which minimizes the energy J_3 in the gradient descent way [36].

More interestingly, it has also been proved [36, Theorem 4] that $E(\text{vec}\,[G_o])^t E(\text{vec}\,[G]) = 2E(\text{vec}\,[G_o])^t E(\text{vec}\,[G_o]) > 0$, where $G_o = (\vec{y}\vec{x}^T - \vec{y}\vec{y}^T M(t))$, $G = (\vec{y}(\vec{x} - \vec{u})^T - (\vec{y} - \vec{y}')\vec{x}^T)$, and vec transforms a matrix into a column vector by stacking the columns of the matrix underneath each other. This result revealed that in the *average sense*, the subspace rule (50) is also an adaptive "down-hill" rule for minimizing the energy function J_3. Moreover, it has also been proved [36, Theorem 3] that all the critical points of $\partial J_3/\partial M = 0$ are saddle points, except for the one whose column vectors span the same subspace as the k principal component vectors $\vec{\phi}_j$, $j = 1, \cdots, k$ span. Therefore, both the rules (50) and (51) will finally reach the same solution that makes J_3 take the global

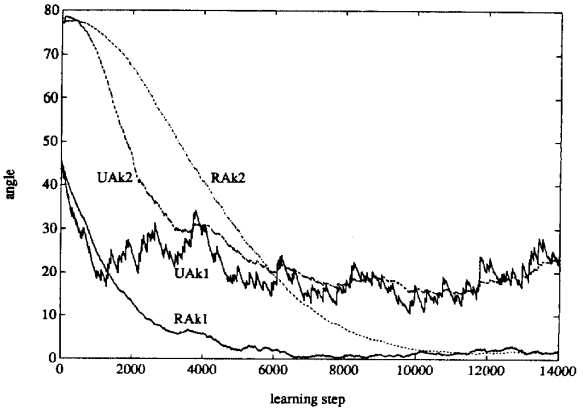


Fig. 6. The learning curves obtained in the comparative experiments for the first two principal component vectors by the robust rule (41) with the combination (44) and its unrobust counterpart GHA. $UAk1$, $UAk2$ denote the learning curves of the angles of $\theta_1 = \text{angle}(\vec{\phi}_1, \vec{\phi}_{p1})$ and $\theta_2 = \text{angle}(\vec{\phi}_2, \vec{\phi}_{p2})$, respectively, obtained by GHA. $RAk1$, $RAk2$ denote the learning curves of the angles obtained by using the robust rule (41).

minimum value. Having the energy function J_3 as a bridge and in parallel to what we did in Section III, we can also generalize the unrobust rules (50) and (51) into robust versions by using the statistical physics approach again. Omitting the details of the derivation, we give the resulting robust generalization of the rule (51) as follows:

$$\vec{M}(t+1) = \vec{M}(t) + \alpha_A(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, M(t)) - \eta)}} \cdot[\vec{y}_i(\vec{x}_i - \vec{u}_i)^T - (\vec{y}_i - \vec{y}_i')\vec{x}_i^T] \tag{53}$$

where $z(\vec{x}_i, M(t)) = \|\vec{x}_i - \vec{u}_i\|^2$.

Similar to (34), we can again observe that the difference of (53) from (51) is that the learning rate $\alpha_A(t)$ has been modified by the following multiplicative factor:

$$\alpha_M(t) = \frac{1}{1 + e^{\beta(z(\vec{x}_i, M(t)) - \eta)}}. \tag{54}$$

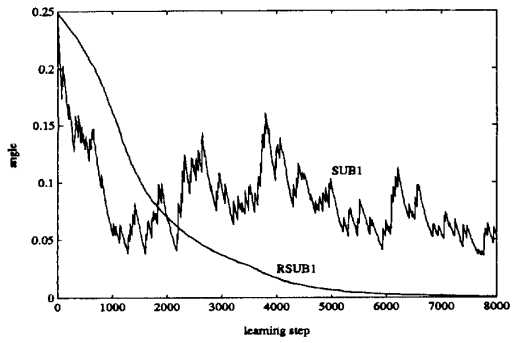
Based on the connection between (50) and the energy function J_3, we can also formally use this $\alpha_M(t)$ to turn the unrobust rule (50) into the following robust version:

$$\vec{M}(t+1) = \vec{M}(t) + \alpha_A(t)\frac{1}{1 + e^{\beta(z(\vec{x}_i, M(t)) - \eta)}}[\vec{y}_i\vec{x}_i^T - \vec{y}_i\vec{y}_i^T M(t)]. \tag{55}$$

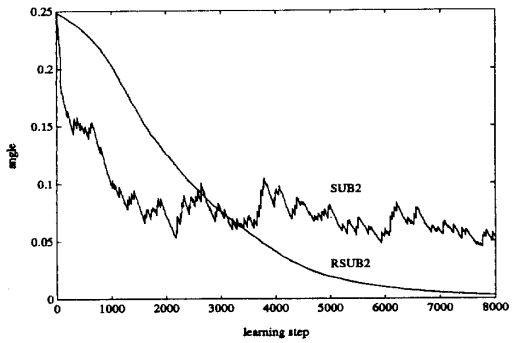
Fig. 7(a), (b) show the results of comparative experiments of using the robust rules and the unrobust rules for solving the two-dimensional principal subspace of the data set used in Figs. 4 and 5. Each learning curve in Fig. 7(a), (b) expresses the change of the residual

$$e_r(t) = \sum_{j=1}^{2}\|\vec{m}_j(t) - \sum_{r=1}^{2}(\vec{m}_j(t)^T\vec{\phi}_{pr})\vec{\phi}_{pr}\|^2 \tag{56}$$

with learning steps. The smaller the residual, the closer the estimated principal subspace to the correct one. In Fig. 7(a), (b), $SUB1$, $SUB2$ denote the unrobust rules (50) and (51), respectively, and $RSUB1$, $RSUB2$ denote the robust rules



(a)



(b)

Fig. 7. The learning curves obtained in the comparative experiments of using the robust rules and the unrobust rules for solving the two-dimensional principal subspace. Each learning curve expresses the change of the residual given by (56). $SUB1$, $SUB2$ denote the unrobust rules (50) and (51), respectively, and $RSUB1$, $RSUB2$ denote the robust rules (55) and (53), respectively.

(55) and (53), respectively. It can be seen that the residuals produced by the unrobust rules $SUB1$, $SUB2$ will fluctuate around a value between 0.05 and 0.1, and will not vanish to zero. However, the residuals produced by the robust rules $RSUB1$, $RSUB2$ approach zero as learning continues. That is, the obtained principal subspace is almost identical to the correct one. We see again that the robust rules really work. In this example, the parameters used are $\alpha_a(t) = 0.003$, $\beta = 2.0$, and $\eta = 0.1$, respectively. In addition, by comparing Fig. 7(a) and (b), we will see that $SUB1$, $RSUB1$ converge faster, but fluctuate more than $SUB2$, $RSUB2$. This is a phenomenon similar to what we discussed at the end of Section V, and the remarks here also apply here.

VI. Further Remarks: Parameter Selection, Minor Component Analysis, and Other Robust PCA Algorithms

First, we make some remarks on the selection of the three parameters α, β, and η which are involved in all the implementations of robust rules proposed in the previous sections. α is the learning rate. The larger the α is, the faster the learning and the bigger the fluctuations in the learning process. β originates from (20) and is the inverse of the temperature $T = 1/\beta$, which determines the sharpness of the Gibbs distribution. This parameter influences robust rules through the modifying multiplicative factor of the learning rate [see (34) and (54)]. The larger β is, the more sensitive this modifying factor will be. η originates from (19), which expresses the amount of penalty contributed to the whole energy when a data point is considered as an outlier point. The larger the η, the heavier the penalty when a data point fails to be considered as a simple point.

In all the experiments in this paper, for simplicity we selected fixed values for α, β, and η (chosen arbitrarily). However, in practice, in order to optimize performance, it is better to vary the three parameters as learning goes on. Usually, 1) α starts at an initial value and decreases as learning proceeds according to the condition given by (7); 2) β starts at a value small enough and then increases with a rate of $O(\ln t)$; 3) η changes according to β: for small β, the modifying factor given by (34) or (54) is not sensitive, and η can be small to reduce the outliers' impact on the learning; for a larger β, the modifying factor is more sensitive, and thus it is better to also increase η to a large value so that the true sample points are not be considered as outliers.

Second, we consider how to modify the proposed robust PCA rules to do robust *minor component analysis (MCA)*, i.e., to find vectors $\vec{\phi}_j$, $j = 1, \cdots, k$ so that $E\{(\vec{\phi}_j^T \vec{x})^2\}$, $j = 1, \cdots, k$ are minimized under the constraints $\vec{\phi}_i^T \vec{\phi}_j = \delta_{ij}$ for $j < i$. The solution vectors $\vec{\phi}_j$, $j = 1, \cdots, k$ are the k orthogonal unit vectors solved by $\Sigma\vec{\phi}_j = \lambda_j \vec{\phi}_j$, $j = 1, \cdots, k$ with λ_j, $j = 1, \cdots, k$ being the k smallest eigenvalues of Σ in ascending order of magnitude. Correspondingly, the one with the smallest eigenvalue is called the minor component vector, and the k vectors are called k-minor component vectors. The problem is encountered in some applications such as curve fitting [37] and dual subspace recognition [38].

As studied in [37], it is not difficult to modify a PCA rule of the type given in (4)–(6) to find the minor component vector. One only needs to replace the learning rate $\alpha_a(t)$ by its negative $-\alpha_a(t)$. However, not every such resulting rule works well. The ones resulting from (4), (6) will diverge in the magnitude $\|\vec{m}(t)\|$, although the direction of $\vec{m}(t)$ will tends to that of the minor component vector. Only the one resulting from (5), i.e.,

$$\vec{m}(t+1) = \vec{m}(t) - \alpha_a(t)\left(\vec{x}y - \frac{\vec{m}(t)}{\vec{m}(t)^T\vec{m}(t)}y^2\right), \quad (57)$$

will converge to the minor component vector both in its direction and magnitude.

We can further show that this rule (57) is actually an *adaptive* or *stochastic approximation* rule for minimizing the following energy function in the gradient descent way:

$$J_4(\vec{m}) = \frac{Ey^2}{\vec{m}^T\vec{m}} = \frac{\vec{m}^T\Sigma\vec{m}}{\vec{m}^T\vec{m}}, \; y = \vec{m}^T\vec{x};$$
$$\text{or } J_4(\vec{m}) = \frac{1}{N}\sum_{i=1}^{N}\frac{\vec{m}^T\vec{x}_i\vec{x}_i^T\vec{m}}{\vec{m}^T\vec{m}}. \quad (58)$$

This point is not difficult to be seen by taking the derivative $\partial J_4/\partial \vec{m}$ on one hand and taking the expectation on (57) on the other hand. Using the energy function J_4 as a bridge, and in parallel to what we did in Section III for obtaining (32), we can also generalize the unrobust rule (57) into a robust version by using the statistical physics approach. Omitting the detailed derivation, we directly give the obtained robust MCA rule as follows:

$$\vec{m}(t+1) = \vec{m}(t) - \alpha_a(t)\frac{1}{1+e^{\beta(z(\vec{x},\vec{m}(t))-\eta)}} \cdot \left(\vec{x}y - \frac{\vec{m}(t)}{\vec{m}(t)^T\vec{m}(t)}y^2\right) \quad (59)$$

where $z(\vec{x}, \vec{m}(t)) = y^2/\vec{m}(t)^T\vec{m}(t)$. Moreover, by a similar derivation to that for getting (41), the rule can be further extended to the following rule for solving the k minor component:

$$\vec{m}_j(t+1) = \vec{m}_j(t) - \alpha_a(t)\frac{1}{1+e^{\beta(z(\vec{x}(j),\vec{m}_j(t))-\eta)}} \cdot \left(\vec{x}(j)y(j) - \frac{\vec{m}_j(t)}{\vec{m}_j(t)^T\vec{m}_j(t)}y(j)^2\right),$$

$$\vec{x}(0) = \vec{x},\ \vec{x}(j+1) = \vec{x}(j) - \sum_{r=1}^{j-1} y(r)\vec{m}_r(t),$$
$$y(j) = \vec{m}_j^T(t)\vec{x}(j). \quad (60)$$

Finally, in the rest of this section, we briefly review the existing robust PCA algorithms in the literature of statistics, and discuss their differences from the rules proposed in this paper. In addition, we will also propose a robust PCA rule which is developed based on the statistical physics approach too, but which has some similarity to some of the algorithms in the literature of statistics.

In contrast to the state of the self-organizing rules for PCA in the neural networks literature, where little attention has been paid to robust rules for resisting outliers, the problem of robust PCA has been studied for years in statistics literature [14], [33], [10], [9], and there already exist several algorithms. These existing algorithms are basically of two types. For the first type, the standard PCA analyzing procedure of (2) and (3) is still implemented, but in parallel to the procedure, some diagnostic statistics (e.g., influence function [9]) or graphical displaying techniques are used to detect and discard outliers from computing the sample covariance matrix S in (3). The performance of this type of algorithm highly depends on the effectiveness of outlier detection; usually, the existing diagnostic statistics are not simple to compute and work well only in some specific situations. For the second type, the robust estimate of sample covariance matrix S^* is pursued by some robust statistical techniques, and then this S^* is used to solve the principal component vectors by (3). There are two usual ways to get S^*. One is to calculate each element of S^* individually by the specific robust estimator for the correlation or covariant coefficient. This way has the disadvantage that the resulting S^* may no longer be semi-positive, and thus affects the solution of (3). The other way is to estimate S^* as a whole, e.g., by

$$\vec{\mu} = \frac{\sum_{i=1}^N w_1(d_i^2)\vec{x}_i}{\sum_{i=1}^N w_1(d_i^2)},$$
$$S^* = \frac{1}{N}\sum_{i=1}^N w_2(d_i^2)(\vec{x}_i - \vec{\mu})(\vec{x}_i - \vec{\mu})^T \quad (61)$$

where $w_1(d_i^2)$, $w_2(d_i^2)$ are a type of commonly used Huber's weighting coefficients [14]. One simple example is given as follows:

$$w_1(d_i^2) = 1, \quad d_i^2 \le k, \quad \text{and} \quad w_2(d_i^2) = w_1(d_i^2)^2/\gamma.$$
$$= k/d_i^2, \quad \text{otherwise} \quad (62)$$

with $d_i^2 = (\vec{x}_i - \vec{\mu})^T S^{*-1}(\vec{x}_i - \vec{\mu})$ and k, γ being some prespecified coefficients.

Like the standard PCA analyzing procedure (2) and (3), the algorithms of both types above are implemented in *batch way*, and their differences from the standard procedure is only that the sample covariance matrix is estimated differently. Thus, these algorithms are obviously different from the self-organizing or adaptive PCA rules currently studied in the literature of neural networks, and from the robust rules proposed in this paper. Specifically, our robust rules are different from these algorithms in at least two aspects. First, for our robust rule, there is no explicit computation for obtaining the sample covariance matrix and for solving eigenequation (3). Instead, the solution vectors are obtained directly by samples fed in adaptively, through step-by-step modifications on those arbitrarily given initial vectors. Second, for our robust rules, no hard decision is needed to detect outliers during the learning process, and outliers are considered implicitly and smoothly through the effective energy given by (23), which in turn leads to the modification of learning rate in the adaptive rules. In contrast, for the algorithms of the above two types, some hard heuristic decision should be made to detect outliers which are then discarded or appropriately weighted in the next step of the computation of the sample covariance matrix.

However, it is interesting that, based on the statistical physics approach, we can also develop an algorithm similar to the above second type. This algorithm consists of the following two stages which are recursively implemented as β is gradually increased to ∞.

1) Approximate the binary field $\{V_i\}$ by its corresponding mean field given by

$$\overline{V}_i(t+1) = \frac{1}{t+1}\frac{1}{1+e^{\beta(z(\vec{x}_i,\vec{m})-\eta)}} + \frac{t}{t+1}\overline{V}_i(t), \quad (63)$$

where initially $\overline{V}_i(0) = 0$.

2) Solve

$$S^{(t+1)}\vec{m}(t+1) = \eta_t\vec{m}(t+1)$$
$$S^{(t+1)} = \frac{1}{N}\sum_i \overline{V}_i(t+1)\vec{x}_i\vec{x}_i^T \quad (64)$$

where η_t, $\vec{m}(t+1)$ are, respectively, the largest eigenvalue and the corresponding eigenvector of the semi-positive defined matrix $S^{(t+1)}$.

The first stage comes from the following derivation. It follows directly from the Gibbs distribution that the conditional probabilities of V_i with respect to the other variables, V_j, $j \neq i$ and $\vec{m}$, are given by

$$P[V_i = 1|V_j, j \neq i, \vec{m}] = e^{-\beta z(\vec{x}_i, \vec{m})} F(V_j, j \neq i, \vec{m}),$$

$$P[V_i = 0|V_j, j \neq i, \vec{m}] = e^{-\beta \eta} F(V_j, j \neq i, \vec{m}), \quad (65)$$

where $F(V_j, j \neq i, \vec{m})$ is independent of V_i.

By normalizing the distributions, we obtain

$$P[V_i = 1|V_j, j \neq i, \vec{m}] = \frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}) - \eta)}} \quad (66)$$

and thus, taking the expectation with respect to the Gibbs distribution, we obtain

$$\overline{V}_i = E\left(\frac{1}{1 + e^{\beta(z(\vec{x}_i, \vec{m}) - \eta)}}\right). \quad (67)$$

The second stage is obtained by minimizing $E(\vec{V}, \vec{m})$ with respect to $\vec{m}$. That is, $\partial E(\vec{V}, \vec{m})/\partial \vec{m} = 0$, which gives

$$\sum_{V_i} \overline{V}_i \frac{\partial z(\vec{x}_i, \vec{m})}{\partial \vec{m}} = 0 \quad (68)$$

since

$$\frac{\partial z(\vec{x}_i, \vec{m})}{\partial \vec{m}} = -\frac{\vec{m}^T \vec{x}_i}{\vec{m}^T \vec{m}} \left(\vec{x}_i - \frac{\vec{m}^T \vec{x}_i}{\vec{m}^T \vec{m}} \vec{m}\right) \quad \text{for } J_1$$

$$\frac{\partial z(\vec{x}_i, \vec{m})}{\partial \vec{m}} = -\vec{m}^T \vec{x}_i \vec{x}_i (2 - \vec{m}^T \vec{m}) + (\vec{m}^t \vec{x}_i)^2 \vec{m} \quad \text{for } J_2. \quad (69)$$

Putting it into (68), we get, respectively,

$$S\vec{m} = \frac{\vec{m}^T S \vec{m}}{\vec{m}^T \vec{m}} \vec{m} \quad (70)$$

$$S\vec{m} = \frac{\vec{m}^T S \vec{m}}{2 - \vec{m}^T \vec{m}} \vec{m} \quad (71)$$

with $S = (1/N) \sum_{i=1}^{N} \overline{V}_i \vec{x}_i \vec{x}_i^T$. Thus, the solution which satisfies (70) and (71) and minimizes $E(\vec{V}, \vec{m})$ is the largest eigenvector of the problem $S\vec{m} = \eta \vec{m}$.

VII. Conclusions

The existing adaptive PCA rules are designed to work on data that have not been spoiled by outliers. In practice, real data often contain some outliers, and usually they are not easy to separate from the data set. These outliers will significantly deteriorate the performances of the existing PCA algorithms. In this paper, we have adapted the statistical physics approach to tackle the problem of robust PCA, and have generalized several commonly used PCA self-organizing rules into robust versions. We have studied in detail various PCA-like tasks such as obtaining the first principal component vector, the first k principal component vectors, and the subspace spanned by the first k vectors directly without solving for each vector individually. For these tasks, we have shown, through a number of comparative experiments, that the robust rules proposed in this paper can resist outliers very well and have improved the performances of the existing PCA rules significantly.

The problems of outliers will also be encountered in supervised learning (for example, in the training of a multilayer perceptron by backpropagation), although, to our knowledge, this has not been discussed in the literature. It seems straightforward to extend our current work to the models of training with teachers.

References

[1] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Networks*, vol. 2, pp. 52–58, 1989.

[2] ——,"Back-propagation and unsupervised learning in linear networks," in Y. Chauvin and D. E. Rumelhart, Eds., *Back-Propagation: Theory, Architecture, and Applications*. Hillsdale, NJ: Erlbaum Associates, 1991.

[3] H. G. Barrow, "Learning receptive fields," in *Proc. 1987 IEEE 1st Annu. Conf. Neural Networks*, vol. 4, 1987, pp. 115–121.

[4] W. Bialek, D. L. Ruderman, and A. Zee, "Optimal sampling of natural images: A design principle for the visual system," in *Advances in Neural Information Processing System 3*, R. P. Lippmann, J. E. Moody, and D. S. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 363–369.

[5] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, pp. 291–294, 1988.

[6] R. Brockett, "Dynamical systems that learn subspaces," in *Mathematical System Theory: The Influence of R. E. Kalman*. Berlin: Springer-Verlag, 1991, pp. 410–420.

[7] Y. Chauvin, "Principal component analysis by gradient descent on a constrained linear Hebbian cell," in *Proc. IEEE Int. Conf. Neural Networks, Vol. I* (Washington, DC). New York: IEEE Press, 1989, pp. 373–380.

[8] G. W. Cottrell, P. W. Munro, and D. Zipser, "Image compression by back-propagation: A demonstration of extentional programming," Tech. Rep. 8702, Inst. Cognitive Sci., UCSD, 1987.

[9] F. Critchley, "Influence in principal components analysis," *Biometrika*, vol. 72, pp. 627–636, 1985.

[10] S. J. Devlin, R. Gnanadesikan, and J. R. Kettenrig, "Robust estimation of dispersion matrices and principal components," *J. Amer. Statist. Ass.*, vol. 76, pp. 354–362, 1981.

[11] P. Foldiak, "Adaptive network for optimal linear feature extraction," in *Proc. IEEE Int. Conf. Neural Networks, Vol. I* (Washington, DC). New York: IEEE Press, 1989, pp. 401–405.

[12] D. Geiger and A. L. Yuille, "A common framework for image segmentation," *Int. J. Comput. Vision*, vol. 6, pp. 227–233, 1991.

[13] K. Hornik and C. M. Kuan, "Convergence analysis of local feature extraction algorithms," *Neural Networks*, vol. 5, , pp. 229–260, 1992.

[14] P. J. Huber, *Robust Statistics*. New York: Wiley, 1981.

[15] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, pp. 79–87, 1991.

[16] D. M. Kammen and A. L. Yuille, "Spontaneous symmetry-breaking energy functions and the emergence of orientation selective cortical cells," *Biol. Cybern.*, vol. 59, pp. 23–31, 1988.

[17] S. Y. Kung, "Adaptive principal component analysis via a new neural learning network," in *Proc. 1990 IEEE Int. Symp. Circuits Syst.* (New Orleans, LA). New York: IEEE Press, 1990, pp. 138–140.

[18] E. Linsker, "From basic network principles to neural architecture," *Proc. Nat. Acad. Sci. USA*, vol. 83, pp. 7508–7512, 8390–8394, 1986.

[19] ——,"Self-organization in a perceptual network," *IEEE Computer*, pp. 105–117, Mar. 1988.

[20] ——,"Design a sensory processing system: What can be learned from principal components analysis?," in *Proc. Int. Joint Conf. Neural Networks*, Vol. II, (Washington, DC). 1990, pp. 291–297.

[21] D. J. C. Mackey and K. D. Miller, "Analysis of Linsker's simulation of Hebbian rules," *Neural Computation*, vol. 2, pp. 173–187, 1990.

[22] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 16, pp. 267–273, 1982.

[23] ——,*Subspace Methods of Pattern Recognition*. Letchworth, England: Research Studies Press, 1983.

[24] E. Oja and J. Karhunen, "On stochastic approximation of eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Appl.*, vol. 106, pp. 69–84, 1985.

[25] E. Oja, "Neural networks, principal components, and subspaces," *Int. J. Neural Syst.*, vol. 1, pp. 61–68, 1989.
[26] E. Oja, H. Ogawa, and J. Wangviwattana, "Learning in nonlinear constrained Hebbian networks," in *Proc. ICANN-91*, Helsinki, Finland, 1991, pp. 385–390.
[27] E. Oja, "Principal components, minor components, and linear neural networks," *Neural Networks*, vol. 5, pp. 927–935, 1992.
[28] J. Rubner and P. Tavan, "A self-organizing network for principal-component analysis," *Europhys. Lett.*, vol. 10, pp. 693–689, 1989.
[29] F. Palmieri and J. Zhu, "A comparison of two eigen-networks," in *Proc. Int. Joint Conf. Neural Networks*, Vol. II, (Seattle, WA). 1991, pp. 193–199.
[30] G. Parisi, *Statistical Field Theory*. Reading, MA: Addison-Wesley, 1988.
[31] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood, and the EM algorithm," *SIAM Rev.*, vol. 26, pp. 195–239, 1984.
[32] J. Rubner and K. Schulten, "Development of feature detectors by self-organization," *Biol. Cybern.*, vol. 62, pp. 193–199, 1990.
[33] F. H. Ruymagaart, "A robust principal component analysis," *J. Multivariate Anal.*, vol. 11, pp. 485–497, 1981.
[34] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feed forward neural network," *Neural Networks*, vol. 2, pp. 459–473, 1989.
[35] R. Williams, "Feature discovery through error-correcting learning," Tech. Rep. 8501, Inst. Cognitive Sci., UCSD, 1985.
[36] L. Xu, "Least MSE reconstruction for self-organization: (II) Further theoretical and experimental studies on one layer nets," in *Proc. Int. Joint Conf. Neural Networks*, Singapore, Nov. 1991, pp. 2368–2373.
[37] L. Xu, E. Oja, and C. Y. Suen, "Modified Hebbian learning for curve and surface fitting," *Neural Networks*, vol. 5, pp. 441–457, 1992.
[38] L. Xu, A. Krzyzak, and E. Oja, "Neural nets for dual subspace pattern recognition," *Int. J. Neural Syst.*, vol. 2, pp. 169–187, 1991.
[39] L. Xu, "Least mean square error reconstruction for self-organizing neural nets," *Neural Networks*, vol. 6, pp. 627–648, 1993.
[40] A. L. Yuille, D. M. Kammen, and D. S. Cohen, "Quadrature and the development of orientation selective cortical cells by Hebb rules," *Biol. Cybern.*, vol. 60, pp. 183–194, 1989.
[41] A. L. Yuille, D. Geiger, and H. H. Bulthoff, "Stereo integration, mean field and psychophysics," *Network*, vol. 2, pp. 423–442, 1991.
[42] A. L. Yuille, "Generalized deformable models, statistical physics, and matching problems," *Neural Computation*, vol. 2, pp. 1–24, 1990.
[43] A. L. Yuille, T. Yang, and D. Geiger, "Robust statistics, transparency and correspondence," Harvard Robotics Lab., Tech. Rep. 90-7, 1990.

Lei Xu (SM'94) received the B.Eng. degree in December 1981 from the Department of Electrical Engineering, Harbin Institute of Technology, and the M.Eng. and the Ph.D. degrees, both on pattern recognition and signal processing, from the Department of Automation, Tsinghua University, China.

In 1988, he became one of ten exceptionally promoted young Associate Professors at Peking University. From 1989 to 1993, he worked at the Department of Information Technology in Lappeenranta, University of Technology, Finland; the Department of Computer Science in Concordia University, Canada; Division of Applied Science at Harvard University, Cambridge, MA; and the Department of Brain and Cognitive Sciences at MIT, Cambridge, MA, as Senior Research Associate, Research Associate, Visiting Scientist and Postdoctoral Associate, respectively. Since September 1993, he has been a Senior Lecturer at the Department of Computer Science at the Chinese University of Hong Kong. Adjunctly, he has been a full Professor of Information Science Center and National Laboratory on Machine Perception at Peking University since September 1992.

Dr. Xu is an author of over 100 academic papers on neural networks, computer vision, signal processing, pattern recognition, and artificial intelligence. He has given invited presentations and lectures at several international conferences on neural networks and many universities, internationally. He is a Technical Program Committee Co-Chairman of the International Conference on Neural Information Processing, 1996, Hong Kong. He was the organizer of two NIPS Postconference Workshops in 1992, 1994, respectively, and served as Session Chair and Program Committee members for several international conferences on neural networks. He is an associate editor of *Neural Networks*, *IEEE Transactions on Neural Networks*, and *International Journal of Neural Systems*. Dr. Xu was a winner of the first Young Teacher Prize by the Chinese National Education Council Fok Ying Tung Education Foundation in 1988. He was also the second of the 10 winners of the Second Beijing Young Scientists' Prize awarded by Beijing Association for Science and Technology in 1988. He was also the first of the five winners of the Excellent Paper Award for young researchers in the 1988 National Conference of the Chinese Automation Society. He is also one co-recipients of a Chinese National Nature Science Award in 1994.

Dr. Xu is a member of the Technical Committee on Neural Networks (TC3) of the International Association for Pattern Recognition, and a vice president of Asian-Pacific Neural Network Assembly. He is also a member of The New York Academy of Sciences, and of Sigma Xi, and The Scientific Research Society.

Alan L. Yuille (M'91) was born in England, with Australian citizenship, in 1955. He received the B.A. degree in mathematics with First Class Honors from Cambridge University in 1976. In 1980 he completed the Ph.D. degree with a Thesis on "Topics in Quantum Gravity" in the Department of Applied Mathematics and Theoretical Physics, Cambridge University. In 1981 and 1982 he did post-doctoral work in physics at the University of Texas at Austin and the Institute for Theoretical Physics at Santa Barbara, CA.

During this period he became interested in theoretical neuroscience. From 1982 to 1986 he held Visiting and Research Scientist appointments at the Artificial Intelligence Laboratory at MIT, Cambridge, MA. In 1986 he joined the Division of Applied Sciences at Harvard University, Cambridge, MA as a lecturer, was appointed Assistant Professor of Computer Science in 1988, and Associate Professor in 1992. His current research interests are computational vision, psychophysics, theoretical and real neural networks, the theory of brain development and learning.

Dr. Yuille has published about 100 academic papers. He has given invited presentations and lectures at several international conferences on neural networks and computer vision, and at a number of USA, Canada and Europe universities. Dr. Yuille served as a Technical Program Committee Co-Chairman of the 1992 International Conference on Neural Information Processing Systems, Denver, CO and as Session Chairs And Program Committee members for several international conferences on neural networks and visual processing. He is an editor of *Journal of Mathematical Imaging and Vision*. He was a winner of the Rayleigh Research Prize in 1979, Honorary Mention of the Marr Prize by ICCV in 1988. He was the Fesler-Lambert Visiting Professor, University of Minnesota in June 1992, and the Visiting Scientist, Isaac Newton Institute of Mathematics in 1993.