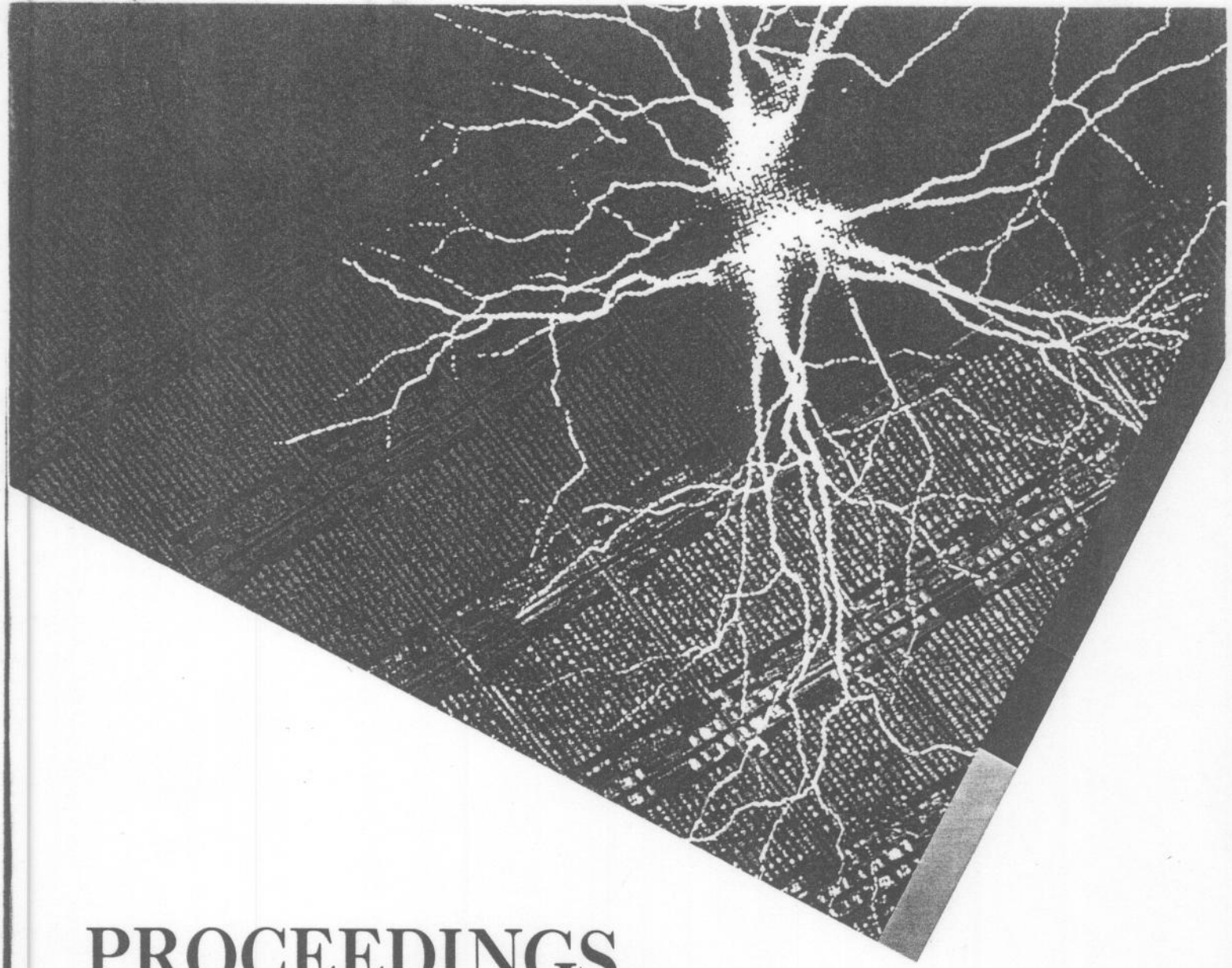


PROCEEDINGS

VOLUME 2 of 3

神經回路網 國際學術大會

International Conference on Neural Information Processing

OCT. 17 (Mon) - 20 (Thur), 1994, SEOUL, KOREA

ICONIP '94

Beyond PCA Learnings: From Linear to Nonlinear and From Global Representation to Local Representation

(Invited Talk)

Lei Xu
Dept. of Computer Science, The Chinese University of Hong Kong
Shatin, Hong Kong, Email: xu@cs.cuhk.hk
and
Information Science Center, Peking University, Beijing

***Abstract*— First, a new *Principal Component Analysis (PCA)* learning theory is proposed, variants of some previous PCA theories are presented, and a concise and systematic summary on PCA theories is tabularly provided. Second, three types of PCA nonlinear extensions are proposed. Particularly the nonlinear extensions of two PCA theories MVNO and LMSER are investigated in detail and and their new properties are revealed, based on which new hierarchical clustering methods are proposed for both data discrimination and data compression/reconstruction. Third, the localized PCA methods are also suggested.**

1 Introduction

A lot of advances have been made on PCA learning type self-organizing networks for the decade since Oja's pioneer work on a single PCA neuron [5]. A long reference list for various PCA networks is given in [14], and a detailed summary on PCA learning theories and recent developments on PCA extension is given in [17]. Due to space limit, we here apologize to not repeat them again.

In the next section of this paper, we first present a new PCA learning theory called MVNO (*Maximum Variation of Normalized Output*), and then give several variants and modifications of the theories proposed in [17]. In addition, a more concise and systematic summary on PCA learning theories will be provided in Table 1. Section 3 proposes three types of PCA nonlinear extensions and investigates the nonlinear extensions of MVNO proposed in section 2 and LMSER proposed earlier in [14] with their new properties revealed. In section 4, three hierarchical clustering algorithms are proposed for unsupervised learning tasks of both discrimination-purposed and compression/reconstruction-purposed. Finally, section 5 suggests that the localized PCA methods can also be used for these learning tasks.

2 PCA Learning Theories: New Results + A Review

We consider a linear network $y = W^t x$ (see the notes under Table 1 for the notations). Here, we propose a new PCA learning theory called *Maximum Variation of Normalized Output (MVNO)*. The idea is that we let $y = DW^t x$ being normalized by $(W^t W)^{-\frac{1}{2}}$ and then maximize the variation of the normalized output y. One cost function for this purpose is given as

$$\max_W J_n, \ J_n = E(y^t y) = tr(E(yy^t)), \ y = (W^t W)^{-\frac{1}{2}} DW^t x. \tag{1}$$

From $E(yy^t) = DW^t \Sigma_x W D(W^t W)^{-1}$ and $\nabla_W J_n = 0$, we can get

$$\Sigma_x W D(W^t W)^{-1} D = W(W^t W)^{-1} DW^t \Sigma_x W D(W^t W)^{-1}$$

By single value decomposition we have $W = \Phi D_w R^t$, where Φ is a $n \times k$ matrix with $\Phi^t \Phi = I$, R is $k \times k$ orthogonal matrix, and D_w is $k \times k$ diagonal matrix. Putting this W in eq.(1), after some derivation we have $\Sigma_x \Phi = \Phi \Lambda_x$ and that $\Lambda_x [R^t D(RD_w^2 R^t)^{-1} D(RD_w^2 R^t) D^{-1}] = [D_w^{-2} R^t D] \Lambda_x$. Since Λ_x is a diagonal, the parts in "[.]" of the both sides should be diagonal. This is possible only when $R = I$, and then $\Lambda_x D = D_w^{-2} D \Lambda_x$ or $D_w = I$. In other words, Φ consist of k eigenvectors of Σ_x. Moreover, we have $J_n = tr(\Lambda_x D^2)$ which arrives its maximum when Φ consists of the eigenvectors that correspond to the k largest eigenvalues of Σ_x with all the other solutions for Φ being saddle points. In summary, eq.(1) performs the true k-PCA.

By gradient ascent, we can implement eq.(1) via either *batch* rule or *on line* rule as given in Table 1. Moreover, eq.(1) can also be extended into $\max_W f(EJ_n)$ without changing performance. A variant of eq.(1) is to let $J_n = E\|y_1 - y_2\|^2$ for $y_i = (W^t W)^{-\frac{1}{2}} DW^t x_i, i = 1, 2$ with x_1, x_2 being i.i.d. variables. It is equivalent to eq.(1) since $E\|y_1 - y_2\|^2 = 2Ey^t y$. Another variant is to replace $tr(E(yy^t))$ by $det(E(yy^t))$.

For a systematical overview, we summarize the existing cost-function-based PCA theories in Table 1.

Best Reconstruction and *Min-Distorted Reflection* are proposed in [17]. Actually, *Best Reconstruction* is a special case of the LMSER (*Least Mean Square Error Reconstruction*) theory proposed in [11, 14]. *Maximum Relative Uncertainty (MRU)* is proposed in [17]. As shown in [14, 17], these theories originally perform only *Principal Subspace Analysis (PSA)*. In the present paper, an extension made is to use $y = DW^t x$ replacing the original $y = W^t x$ with diagonal elements of D being positive only. Along a line of thought similar to MVNO, it can be shown that the modification on MRU in Table 1 will perform the

true k-PCA as lomg as $D \neq I$. One disadvantage that still remains is that the theory can only be implemented in *batch* way but not in *on line* way. Another extension to is to use

$$\max_W \rho_r, \ \rho_r = tr[E(yy^t)]/det[E(\eta\eta^t)] = E(tr(yy^t))/det(W^tW). \tag{2}$$

We can also prove that this maximization performs the true k-PCA. However, it can be implemented by either *batch* or *on line* rule as given in Table 1.

Maximum Variation by Gradient flow in $O(n,k)$ has been studied by [1, 14, 8]. The one given in Table 1 is proposed by [14] which is an extension of gradient flow [1] in $O(n,n)$ to $O(n,k)$ with $k < n$. *Maximum Combined Variation* is partly presented in [17] but with some new modifications here. It can be shown that the theory performs PSA and the true k PCA under the condition that the constant $c > 0$ is appropriately given such that the diagonal matrices $I + (-c)^{\frac{-1}{2n-1}} det^{\frac{2}{2n-1}}(D^2\Lambda_x)(D^2\Lambda_x)^{-1}$ or $D_w^2 = [1 + (-c)^{\frac{-1}{2n-1}}]I$ have positive diagonal elements only. For example, for $n = 1$ we need $c > D^2\lambda_{max}$ or simply $c > 1$.

Table 1: THEORIES AND LEARNING RULES

Theories	Functions		Learning Rules		Refs
	PSA	*k-PCA*	*Batch*	*On-line*	
Best Reconstruction					Xu
· min $_W E(e_r)$, $e_r = \|x - WW^tx\|^2$	Yes	No	$\Delta W = \alpha\frac{\partial E(e_r)}{\partial W}$	$\Delta W = \alpha\frac{\partial e_r}{\partial W}$	(91,93)
· min $_W f(E(e_r))$	Yes	No	$\Delta W = \alpha\frac{\partial f(E(e_r))}{\partial W}$	No	This
Min-Distorted Reflection					paper
· min $_W \sum_0^k E(e_r)$, $e_r = \|u_{i+1} - u_i\|^2$	Yes	No	$\Delta W =$	$\Delta W =$	Xu
or $e_r = \|u_{i+1} - x\|^2$, $u_{i+1} = WW^tu_i$	Yes	No	$\alpha\sum_0^k \frac{\partial E(e_r)}{\partial W}$	$\alpha\sum_0^k \frac{\partial e_r}{\partial W}$	(94a)
Max. Variation of Normalized Output					
$y = (W^tW)^{-\frac{1}{2}}DW^tx$	Yes	Yes		$\Delta W = \alpha(xy'^tD -$	
· max $_W E(J_n)$, $J_n = y^ty$	$D = I$	$D \neq I$		$Wy'y'^t)$	
or $J_n = \|y_1 - y_2\|^2$	Yes	Yes		$y' = (W^tW)^{-\frac{1}{2}}y$	This
y_1, y_2 (x_1, x_2) are i.i.d.			$\Delta W =$		
· max $_W f(E(J_n))$	Yes	Yes	$\alpha\frac{\partial f(E(J_n))}{\partial W}$	No	paper
· max $_W f(J_d)$, $J_d = det(E(yy^t))$	Yes	Yes	$\Delta W = \alpha\frac{\partial f(J_d)}{\partial W}$	No	
Max-Relative Uncertainty					
$y = DW^tx, \eta = W^t\xi$, ξ from $N(0,I)$				$\Delta W = \alpha[xy^tD -$	This
· max $_W \rho_r$	Yes	Yes	$\Delta W =$	$\frac{\|y\|^2}{det(W^tW)} \times$	paper
$\rho_r = \frac{tr[E(yy^t)]}{det[E(\eta\eta^t)]}$	$D = I$	$D \neq I$	$\alpha\frac{\partial \rho_r}{\partial W}$	$W(W^tW)^{-1}]$	and
· max $_W f(\rho_r)$	Yes	Yes	$\Delta W =$	No	
or $\rho_r = \frac{f(det(E(yy^t)))}{g(det(\eta\eta^t))}$	Yes	Yes	$\alpha\frac{\partial f(\rho_r)}{\partial W}$	No	
or $\rho_r = f(-E\ln p(y)) - g(-E\ln p(\eta))$	Yes	Yes		No	Xu
$p(.)$ is Gaussian					(94a)
Maximum Variation with Lagrange					
· max $_W J_L$, s.t. $W^tW = I$	N/A	Yes	Solving	No	Fuk-
$J_L = tr(E[yy^t]), y = W^tx$			$\Sigma_xW = \Lambda W$		naga
or $J_L = det[E(yy^t)]$	N/A	Yes	$\Sigma_x = E(xx^t)$	No	(72)
or $J_L = -E\ln p(W^tx)$	N/A	Yes		No	
x from Gaussian					This
· max $_W f(J_L)$	N/A	Yes		No	paper
Maximum Variation by					
Gradient flow in $O(n,k)$					Brock-
· max $_W E[tr(yy^t)]$	Yes	Yes	$\Delta W = \alpha(\Sigma_xWD$	$\Delta W = \alpha(xy^tD$	ett(89)
$y = DW^tx$	$D = I$	$D \neq I$	$-WDW^t\Sigma_xW)$	$-WDyy^t)$	Xu(93) Oja(93)
Maximum Combined Variation					
· max $_W J_M$				$\Delta W = \alpha[xy^tD -$	This
$J_M = tr(\Sigma_y) + cJ_w^2$	Yes	Yes	$\Delta W =$	$cJ_w^2W(W^tW - I)^{-1}]$	
or $J_M = det[\Sigma_y] + cJ_w^2$	$D = I$	$D \neq I$	$\alpha\frac{\partial J_M}{\partial W}$	No	paper
$\Sigma_y = E(yy^t), y = DW^tx$	Yes	Yes		No	
$J_w = det(W^tW - I)$, $c > 0$ is					
a suitable constant					

(i) x, y are n, k ($k > 1$) dimensional vectors respectively and W is an $n \times k$ weight matrix. Without losing generality, it is assumed $E(x) = 0$; x should be subtracted by $E(x)$. When $k = 1$ (i.e., W is a vector w), all the theories reduces to PCA, i.e., they let w to be the first PC of x. Moreover, both MNVO and MRU reduce to identically max $_w tr(w^tExx^tw^t)/w^tw$ studied by [6].
(ii) $f(r), g(r)$ may be same or different. They are any positive and monotonously increasing differentiable functions for $r \geq 0$ (may need to satisfy some mild condition sometimes); e.g, $f(r) = r^p, g(r) = r^q, p \geq 1, q \geq 1$.
(iii) α is a given learning stepsize. D is a given $k \times k$ diagonal matrix with their elements being different.

3 PCA Nonlinear Extensions and Their Favorable Properties

Oja (1991) proposed several nonlinear Hebbian learning rules and demonstrated via experiments that nonlinearity can let the learning resist strong noises or outerlier[7]. In the same period, the present author also proposed nonlinear LMSER rule[11, 14]. It has been shown that the introduction of sigmoid function to linear units can automatically break the symmetry of the homogeneous networks with the behaviors similar to performing the true k-PCA. Two years later, this nonlinear LMSER rule has been applied to signal representation and separation with intersting results[3]. Another type of PCA nonlinear extension is given in [12, 13], where nonlinear factor is introduced into controlling the learning rate of modified Hebbian rules for robust curve fitting and robust PCA. Moreover, the idea of extending Hebbian learning to higher order curve fitting has also been proposed in [12]. Later, this idea has been further

turned into high order Hebbian learning by [10]. Recently, studies on PCA nonlinear extensions are becoming quite interested in the literature, a recent summary is given in [17].

Here, Table 2 systemically proposes three types of extensions for the theories given in Table 1. Type I & II are obtained by extending two main components in these theories, Type III combines the two. In Table 2, only the cases for *Best Reconstruction* (or LMSER) and MVNO are directly given as examples. But it is straightforward to write down the corresponding extensions for the rest of theories in Table 1.

Table 2: PCA NONLINEAR EXTENSIONS

Type I	Type II	Type III
Linear transform $W^t x$ or $DW^t x$ Replaced by nonlinear $S(x): R^n \to R^k$ $S(x) = [s_1(x), \cdots, s_k(x)]$ Sigmoid $s_i(x) = s(w_i^t x)$, $W = [w_1, \cdots, w_k]$ $s(w_i^t x) = \frac{1-exp(-2\beta w_i^t x)}{1+exp(-2\beta w_i^t x)}$ Polynomial $s_i(x)$ is a polynomial of x e.g., $s_i(x) = c_0 + \sum_j c_j x_j + \sum_{p,q} c_{p,q} x_p x_q$	Square norm $\|\cdot\|^2$ Replaced by nonlinear $\Psi(z): R^r \to [0, +\infty)$ $r = n, k$ for $z = x, y$ L_p norm $\Psi(z) = (\sum_{j=1}^r \|z_i\|^p)^{1/p}$ $z = [z_1, \cdots, z_r]$, $p > 0$ Robust $\Psi(y) = \|y\|^2$ for $\|y\|^2 \le C$ $\Psi(y) = C$ for $\|y\|^2 > C$	Linear transform $W^t x$ or $DW^t x$ Replaced by nonlinear $S(x): R^n \to R^k$ and also Square norm $\|\cdot\|^2$ Replaced by nonlinear $\Psi(z): R^r \to [0, +\infty)$
LMSER $\min_W E\|x - WS(W^t x)\|^2$	*LMSER* $\min_W E\Psi(x - WW^t x)$	*LMSER* $\min_W E\Psi(x - WS(W^t x))$
MVNO $\min_W E\|y_s\|^2$, $y_s = (W^t W)^{-\frac{1}{2}} S(W^t x)$	*MVNO* $\min_W E\Psi(y)$, $y = (W^t W)^{-\frac{1}{2}} DW^t x$	*MVNO* $\min_W E\Psi(y_s)$

We focus on Type I to investigate the consequences of introducing nonlinearity. We consider the cases that $S(x)$ is sigmoid given in Table 2. It follows from Fig.(1) that the nonlinearity increases with β. To see how $s(.)$ affects the results, here we first propose a variant of MVNO for a single unit

$$\max_w Es^2(w^t x/\|w\|), \ \textit{with on line rule} \ \Delta w = (\alpha/\|w\|)s(y)s'(y)[x - (w/\|w\|)y], \ \ y = w^t x/\|w\| \quad (3)$$

Observing Fig(2), we know that y denotes the distance of x to line $w^t x = 0$. When $s(.)$ is linear, the larger the distance is, the larger its role is in the square function s^2. However, with the increasing of sigmoid nonlinearity, as shown in Fig.(2), most of samples with certain distances away from $w^t x = 0$ have greatly reduced their contributions to $s^2(.)$ (to a small constant 1); while only those samples very near $w^t x = 0$ are still counted as they are in the linear case. Observing also Fig.(3), where the solid and dashed lines denote respectively the direction of solution w and the corresponding line $w^t x = 0$, we get

Argument 1 *The sigmoid nonlinearity shrinks the shape of data cloud by greatly discounting the samples far away from $w^t x = 0$ or from 0. The larger the nonlinearity is, the significance the shape shrunk. As β changes its value, the solution w also changes continuously.*

As shown in Fig.(4), for the two data clouds with their overall mean being around zero, we can observe that sigmoid nonlinearity focus on these samples in order to find a boundary $w^t x = 0$ which lets $s^2(w^t x/\|w\|) \approx 1$ for as many as possible samples. That is, it seeks a narrow band that centers at $w^t x = 0$ such that the number of samples fallen within the band are smallest. The width of the band is decided by β. However, as shown by the two pairs of shorter line segments in Fig.(5), such a boundary seeking property may not work well for two data clouds with their total center not at zero, since $max\ Es^2(w^t x/\|w\|)$ only drives w rotating around zero. To overcome this problem, we modify eq.(3) into

$$\max_{w,a} Es^2(y), \ \Delta w = \gamma[x - \mu - (w/\|w\|)y], \ \Delta\mu = -\gamma w, \gamma = (\alpha/\|w\|)s(y)s'(y), \ y = w^t(x - \mu)/\|w\| \quad (4)$$

It is better to initialize by the mean vector of the data (the same for μ in eqs.(5)(5)(9) given later).

In Fig.(5), the long dashed line segment denotes the line $w^t x = 0$ found by eq.(4), with the direction of its w given by a solid half segment. In summary, we have

Argument 2 *For two data clouds separated by a gap (it may contains small amount mixed samples), sigmoid nonlinearity makes the maximization eq.(4) focus on the boundary samples of the two clouds in order to find a boundary $w^t(x - \mu) = 0$ such that within a small width of it there are as few as possible samples. In other words, the $w^t(x - \mu) = 0$ bestly separates the two clouds.*

This new discovery of nonlinear PCA type learning is very interesting and supplies a basis for our new clustering methods in the subsequent section. It also provides for interpreting nonlinear Hebbian learning a new perspective far different from the very recent statistical interpretation given by [9].

It is not difficult to see that an equivalent but simplified variants of eqs.(3)(4) can be

$$\max_w Es^2(w^t x), \ s.t. \ \|w\| = 1, \ \textit{with on line rule} \ \Delta w = \alpha s(y)s'(y)[x - wy], \ \ y = w^t x,$$
$$\max_{w,a} Es^2(y), \ s.t. \ \|w\| = 1, \ \Delta w = \gamma[x - \mu - wy], \ \Delta\mu = -\gamma w, \gamma = \alpha s(y)s'(y), \ y = w^t(x - \mu). \quad (5)$$

For the nonlinear MVNO given in Table 2, the counterparts of eqs.(3)(4) are given by

$$\max_w Es^2(w^t x)/w^t w, \ \Delta w = (\alpha s(y)/w^t w)[s'(y)x - (s(y)/w^t w)w], \ y = w^t x$$

$$\max_{w,a} Es^2(w^t(x-\mu))/w^tw,\ \Delta w = \gamma[s'(y)(x-\mu) - (s(y)/w^tw)w], \Delta\mu = -\gamma w,$$
$$y = w^t(x-\mu),\ \gamma = \alpha s(y)s'(y) \tag{6}$$

For linear $s(.)$, eq.(6) is exactly the same as eqs.(3)(4). When $s(.)$ is nonlinear, the results are different. However, our experiments have shown that their qualitative properties are similar and that Argument 1& 2 holds too. Rewriting $s(w^tx)$ as $s(\beta' w^tx/\|w\|)$ with $\beta' = \|w\|$, we observe that MVNO tries not only to locate the direction of line $w^tx = 0$, but also to adjust the scale of the band width shown in Fig.(2) by modifying $\beta' = \|w\|$ even when β for $s(.)$ is fixed. This may be a favorable property.

Next we turn to observe, from the viewpoint of data reconstruction, how the nonlinear $s(.)$ acts. We consider the single unit case for nonlinear LMSER given in Table 2, its learning rule is firstly proposed in [11, 14] and is repeated below

$$\min_w E\|x - ws(w^tx)\|^2,\ \Delta w = \alpha[z(x - zw) + s'(w^tx)(w^tx - zw^tw)x],\ z = s(w^tx), \tag{7}$$

The results of the linear PCA, nonlinear LMSER eq.(7) and nonlinear MVNO eq.(6) are shown in Fig.(6). The longest dashed line is the principal component direction found by the linear PCA. The solid lines denote the directions of w found by eq.(7) and eq.(6), and the dash-dotted lines denote the corresponding lines $w^tx = 0$. From this figure, first we can observe that the results of linear PCA, nonlinear LMSER, nonlinear MVNO are different. This confirms our previous analyses in [17] that PCA nonlinear extensions become different although they perform the same in the linear cases. Second, by counting the average reconstruction error $x - ww^tx$ for the linear PCA and $x - ws(w^tx)$ for nonlinear LMSER and MVNO, we get the error of 6.12, 19.07 and 22.93 respectively. The results seem to suggest that linear PCA is the best in the reconstruction error, while the nonlinear LMSER and MVNO are not good choices.

However, the conclusion will become considerably different if we examine the whole data-compression/reconstruction process. The first step is data-compression which produces the compressed signals w^tx or $s(w^tx)$ that are to be encoded for transmission. The second step is to reconstruct the signals at the receiving end by ww^tx or $ws(w^tx)$. For the second step, we hope that the reconstruction error is as small as possible. But for the first step, we hope that the coding bits for w^tx or $s(w^tx)$ are as small as possible to reduce the transmission time and cost. In Figs.(7)(8)(9), the x-axis denotes each of the 1000 data points in Fig.(6). The y-axis gives the value of w^tx or $s(w^tx)$ for each data point. For the linear PCA, the dynamic range of w^tx is quite large and thus a lot of bits are needed to encode each data point. However, $s(w^tx)$ almost only takes value either $+1$ or -1 by nonlinear LMSER and MVNO (denoted by Vmax in the figure). That is, only one bit is needed to encode each data point in the most cases; or in other words, a dipole $\{-w, w\}$ is used such that each data point is approximated by either w or $-w$. If we call the average of the ratio $\frac{\textit{the bits to encode } x}{\textit{the bits to encode } w^tx \textit{ or } s(w^tx)}$ as the data compression rate, we can use the ratio of this data compression rate to the reconstruction error as an index to measure the overall quality of a data-compression/reconstruction process. Based on the above analysis and our current experimental results, we further propose:

Argument 3 *The sigmoid nonlinearity can significantly (probably by magnitudes) improve the ratio of the data compression rate to the reconstruction error. The nonlinear LMSER is better than the nonlinear MVNO, and is probably the best one among all the possible PCA nonlinear extensions.*

This suggests a new technique for data-compression/reconstruction, which will be roughly described in the next section and will be studied in detail in a separated paper.

Now we are ready to move to the cases of multiple units. The *on line* learning rules for the nonlinear MVNO and LMSER in Table 2 are given by

$$\Delta W = \alpha[xz^t\mathcal{D} - Wzz^t],\ z = (W^tW)^{-1}S(W^tx),\ \mathcal{D} = diag[s_1'(y_1), \cdots, s_k'(y_k)],$$
$$\Delta W = \alpha[(x - Wz)z^t + x(x - Wz)^tW\mathcal{D}], \tag{8}$$

For the data with nonzero mean, they can be extended into

$$\Delta W = \alpha[(x-\mu)z^t\mathcal{D} - Wzz^t],\ \Delta\mu = -W\mathcal{D}z,\ z = (W^tW)^{-1}S(W^t(x-\mu)),\ \mathcal{D} = diag[s_1', \cdots, s_k'],$$
$$\Delta W = \alpha[(x-\mu-Wz)z^t + (x-\mu)(x-\mu-Wz)^tW\mathcal{D}],\ \Delta\mu = (I - W\mathcal{D}W^t)(x - WS(W^tx)), \tag{9}$$

The above MVNO rules can also be simplified into variants corresponding to eq.(5) by simply letting $z = S(W^tx), z = S(W^t(x-\mu))$ in eqs.(8)(9). In addition, we can also extend eq.(3) into $\max_W ES^t((W^tW)^{-\frac{1}{2}}W^tx)S((W^tW)^{-\frac{1}{2}}W^tx)$ for multiple units.

Fig.(10) gives an example of using the two rules in eq.(8) for the cases of two units. The two solid lines are the directions of w_1, w_2 obtained by MVNO (denoted by Vmax in the figure), while the dashed lines are the two vectors w_1, w_2 in their real length obtained by LMSER. In Fig.(11), the solid lines and dashed lines still correspond MVNO and LMSER respectively, but now the lines are the boundaries $w_1^tx = 0, w_2^tx = 0$. The result again confirms our previous analyses in [17] that PCA nonlinear extensions become different although they perform the same in the linear cases. More interestingly, after a while of observation, we can find that the experiment supports our following arguments.

Argument 4 *For a data set with zero mean, the nonlinear MVNO rule in eq.(8) finds k boundaries acrossing at zero in order to divide the data into $2k$ parts that are both as separately as possible and as*

equal as possible. For a data set with nonzero mean, the nonlinear MVNO rule in eq.(9) finds k boundaries acrossing at μ in order to divide the data into 2k parts that are both as separately as possible and as equal as possible.

Argument 5 *For a data set with zero mean, the nonlinear LMSER rule in eq.(8) finds k vectors $w_1, \cdots, w_k$ such that each data point x can be labeled by a k-bit binary number $b_1 b_2 \cdots b_k$ ($b_i = -1$ or 1) and well approximated by the linear sum of $\sum_i b_i w_i$. For a data set with nonzero mean, the nonlinear LMSER rule in eq.(9) finds k vectors $w_1, \cdots, w_k$ such that each data point x can be labeled by a k-bit binary number $b_1 b_2 \cdots b_k$ ($b_i = -1$ or 1) and well approximated by the linear sum of $\mu + \sum_i b_i w_i$.*

It is interesting to compare this data representation scheme with the classical vector quantization(VQ) method. For VQ, when k codebooks are used, each of them is directly used to approximate a data point; that is, for a set of data there are in total only k different representations. However, for our above scheme, $k+1$ codebooks (including μ) are used by combination, and there are in total 2^k different representations for a data set. So, our scheme is much more powerful. We can expect to get a great ratio of the data compression rate to the reconstruction error if it is used for data-compression/reconstruction.

4 New Clustering Algorithms for Data Discrimination and Representation

Based on the arguments in section 3, we can design several unsupervised algorithms for the purposes of both data discrimination and representation.

The straight way for unsupervised data discrimination follows from Argument 4: *we use the nonlinear MVNO in eq.(9) to classify data into 2k clusters,* as shown in Fig.(11).

The straight way for data-compression/reconstruction follows from Argument 5: *we use the nonlinear LMSER in eq.(9) to get $k+1$ codebooks $w_1, \cdots, w_k$ and μ, and then encode each data point x by $\mu + S(W^t(x-\mu))$ or quantize $S(W^t(x-\mu))$ into a k bit binary digit $b_1 b_2 \cdots b_k$. Finally, we reconstruct the data by $\mu + WS(W^t(x-\mu))$ or $\mu + \sum_i b_i w_i$.*

In Table 3, we propose three hierarchically-structured clustering algorithms. The basis idea behind the algorithms is the same—building a binary tree T by sequently dividing a set into two subsets. That is, for the current data set D_n we use one single-unit-rule, like eqs.(3) (4) (5) & (6) or even linear PCA rule, to obtain vectors w_n, μ_n, and check if the values of e_1^n, e_2^n, e_3^n are above some prespecified thresholds. If yes, we divide D_n by the boundary $w_n^t(x-\mu_n) = 0$ into two subsets D_n^1, D_n^2. In turn, we repeat the same procedure on D_n^1, D_n^2, until the tree stops its growing. The variables e_1^n, e_2^n, e_3^n are the current reconstruction error or discrimination measure given by

$$e_1^n = \sum_{x \in D_n} \|x - w_n w_n^t (x-\mu_n)\|^2, \quad e_2^n = \sum_{x \in D_n} \|x - w_n S(w_n^t (x-\mu_n))\|^2,$$

$$e_3^n = \frac{tr[Var_{w_n^t(x-\mu_n)>0}(x \in D_n) + Var_{w_n^t(x-\mu_n)>0}(x \in D_n)]}{N_n \|E_{w_n^t(x-\mu_n)>0}(x \in D_n) - E_{w_n^t(x-\mu_n)<0}(x \in D_n)\|^2}, \quad N_n \text{ is a number of samples in } D_n. \tag{10}$$

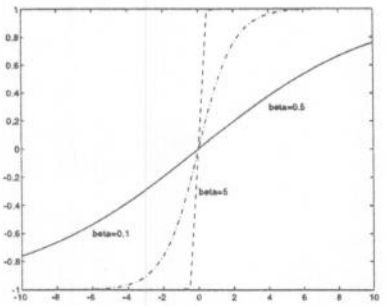
$E_{w_n^t(x-\mu_n)>0}(x \in D_n)$ is the mean vector of samples with $w_n^t(x-\mu_n) > 0$ in D_n, Var denotes their covariance matrix.

Fig.(12) provides a simple example for demonstrating the advantage of Algorithm I. Here, the first linear

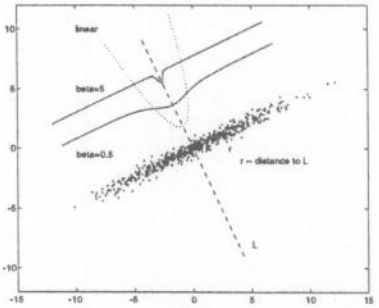
Table 3: Three Hierarchical Clustering Algorithms

		Algorithm I (for data compression)	Algorithm II (for data compression)	Algorithm III (for data discrimination)
	Initial.	Put the data set D in **open** with idex I=0. Presepecify thresholds e_1^t, e_2^t, e_3^t, and let T=null		
TRAINING	Step 1	If **open**=empty, goto Step 3; otherwise, take the 1st element out of **open**, denote it by n with D_n and $I(n)$		
		find the mean μ_n of D_n find the linear 1st PC of D_n as w_n	use one unit nonlinear LMSER in eq.(9) to get its w, μ as w_n, μ_n	use one unit nonlinear MVNO in either eq.(4) or eq.(5) or eq.(6) to get its w, μ as w_n, μ_n
		generate a node n in T at the index $I(n)$, attached with a tuple $[D_n, I(n), w_n, \mu_n]$		
	Step 2	get e_n^1 by eq.(10); if $e_n^1 \le e_1^t$, goto Step 1;	get e_n^2 by eq.(10); if $e_n^2 \le e_2^t$ goto Step 1;	get e_n^3 by eq.(10); if $e_n^3 \le e_3^t$ goto Step 1;
		otherwise, divide D_n into D_n^1 and D_n^2 by $w_n^t(x-\mu_n) = 0$, i.e., $D_n^1 = \{x \in D_n, w_n^t(x-\mu_n) \ge 0\}$ and $D_n^2 = \{x \in D_n, w_n^t(x-\mu_n) < 0\}$ get new indexes by cascading $I(n_1) = I(n)0, I(n_2) = I(n)1$, and put two elements $[D_n^1, I(n_1)]$ and $[D_n^1, I(n_1)]$ at the end of **open** list; then go to Step 1		
	Step 3	The current tree T is used as output, the training stops		
CLASSIFING	ENCODING	Input a data point x, starts from the root node of T, repeat the procedure below: *Suppose that the current node is n with index $I(n)$ and the attached vectors w_n, μ_n,* *classify x to its son with $I(n_1) = I(n)0$ if $w_n^t(x-\mu_n) \ge 0$, otherwise to its son with $I(n_2) = I(n)1$* The procedure is repeated until x is finally classified to a leaf node n_f of T with $I(n_f) = i_1 i_2 \cdots i_k$ and the attached vectors w_f, μ_f. For Algorithm III, this $i_1 i_2 \cdots i_k$ is the resulted class label for x. For Algorithm I & II, after the tree T is pre-sent to the receiving end as the codebook, the $i_1 i_2 \cdots i_k$ plus $s_f = s(w_f^t(x-\mu_f))$ can be used as the code for x for transmission, then at the receiving end, we use $i_1 i_2 \cdots i_k$ to find the corresponding node at the codebook T and get the corresponding w, μ; finally we reconstruct it by $\mu + w s_f$.		

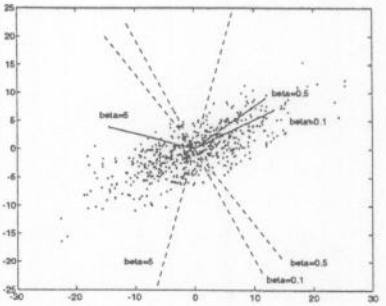
1. T is a tree which grows from the root as the algorithm goes on. A node n at the depth d in the tree is indexed by $I(n) = 0 i_1 i_2 \cdots i_d$ with $i_j = 0$ or 1. Its two sons have indexes $0 i_1 i_2 \cdots i_d i_{d+1}$, $i_{d+1} = 0, 1$. Each node n is attached by a tuple $[I(n), w_n, \mu_n]$.
2. **open** is a list. Each elemnent in the list corresponds to a node to be grown latter. It consists of a tuple $[D_n, I(n)]$.



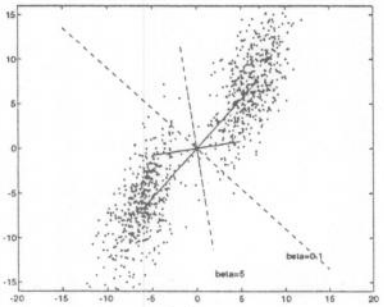
(1)



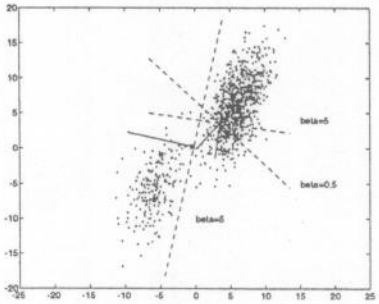
(2)



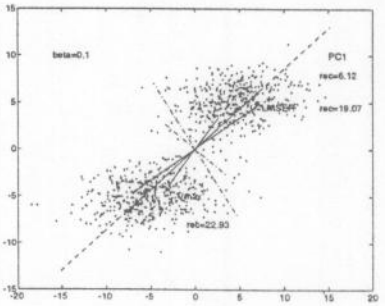
(3)



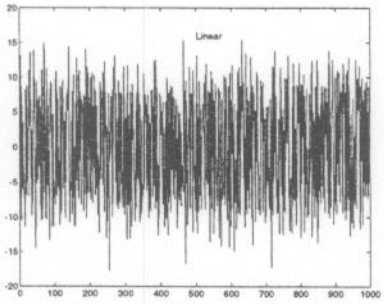
(4)



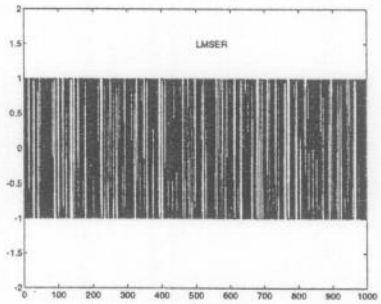
(5)



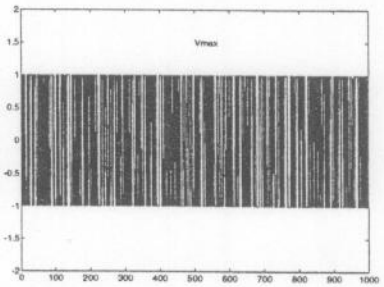
(6)



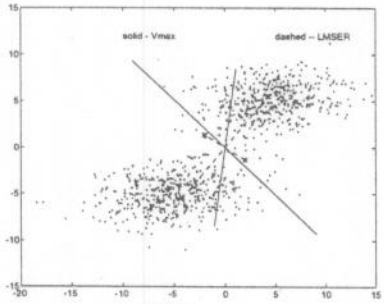
(7)



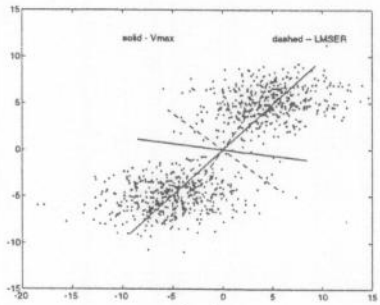
(8)



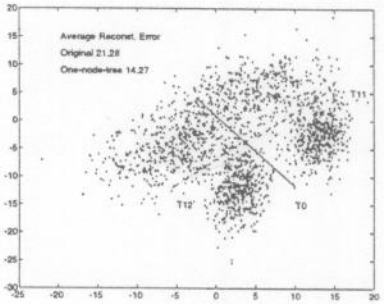
(9)



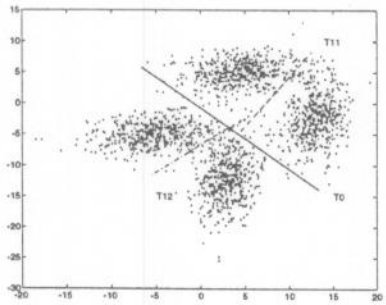
(10)



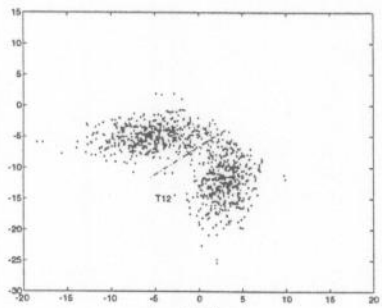
(11)



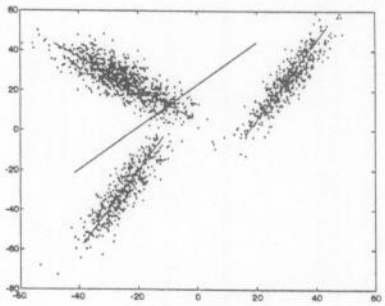
(12)



(13)



(14)



(15)

FIGURES

principal component (PC) vector is computed. The PC w_0 on the whole set is firstly get such that the boundary T_0: $w_0^t(x-\mu_0)=0$ divides the set into two subsets; then the PC on each decides the boundaries T_{11}, T_{12}. As a result, we get an one-node-tree with two leaves. The average reconstruction error is 21.28 if only w_0, μ_0 are used for representing the data set; however it reduces considerably into 14.27 if the one-node-tree is used to represent the data set.

Figs.(13)(14) provides a simple example for demonstrating Algorithm III with eq.(4) used in Step 1. First, at the root of the tree the boundary T_0 is found to separate the data into two parts as its two sons (one of them is more clearly shown in Fig.(14)); then the two sons are further well separated by other two boundaries T_{11}, T_{12}. As a result, the data has been divided into four well separated clusters.

5 Localized PCA

As shown in Fig.(15), when a data set consists of several clusters, it is not appropriate to still globally regard the data as a whole by only using a global PC to represent the set. The long solid line in Fig.(15) denotes the global PC, and obviously it will give a large reconstruction error. So, we should pay more attention on the local structures of the data. Actually, the hierarchical algorithms given in Section 4 are a kind of efforts along a same direction.

Another alternative effort is to first make clustering analysis on data set to separate it into several clusters, and then to represent each cluster by its mean and the PC vectors. This method can be regarded as a good combination of the classical clustering-based VQ and PCA, thus will improve the performances of both. This idea was first suggested in [16] and in the talk for [15] given on WCNN'93-Portland without details. Some similar but different local PCA methods are recently given in [18][4].

This method consists of two separated steps. By the **first step**, with an assumed known number K of clusters we use the finite Gaussian mixture

$$P(x|\Theta)=\sum_{j=1}^{K}\alpha_j P(x|\mu_j,\Sigma_j),\quad P(x|\mu_j,\Sigma_j)=\frac{e^{-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)}}{\sqrt{2\pi|\Sigma_j|}},\quad \alpha_j\geq 0,\ \sum_{j=1}^{K}\alpha_j=1.$$

to model the data set, and use the the following EM iterative algorithm [15]

$$h_j^{(k)}(t)=\alpha_j^{(k)}P(x^{(t)}|\mu_j^{(k)},\Sigma_j^{(k)})/\sum_{i=1}^{K}\alpha_i^{(k)}P(x^{(t)}|\mu_i^{(k)},\Sigma_i^{(k)}),\quad \alpha_j^{(k+1)}=\frac{n_j^{(k)}}{N},\quad n_j^{(k)}=\sum_{t=1}^{N}h_j^{(k)}(t)$$
$$\mu_j^{(k+1)}=\frac{1}{n_j^{(k)}}\sum_{t=1}^{N}h_j^{(k)}(t)x^{(t)},\quad \Sigma_j^{(k+1)}=\frac{1}{n_j^{(k)}}\sum_{t=1}^{N}h_j^{(k)}(t)[x^{(t)}-\mu_j^{(k)}][x^{(t)}-\mu_j^{(k)}]^T.$$

to classify the data into clusters by $h_j(t)$ and to get the mean μ_j and covariance Σ_j of each cluster. The **second step** is to solve one or several PCs of each cluster either simply by eigen-analysis $\Sigma_j W_j = W_j\Lambda_j$ of each covariance or by one of nonlinear rules proposed in the previous sections.

The above proposed first step has a advantage over the use of k-means algorithm for clustering as suggested by [4]. The k-mean is based on the assumption that each cluster has the same covariance matrix, which will seriously distort the real shape of each cluster and give incorrect PCs for each cluster.

The results of the localized PCA can be either used for unsupervised classification for subspace representation of each classifier or distributed data encoding/reconstruction.

Another different way for localized PCA is to solve the μ_j, W_j of each cluster by maximizing:

$$J=-\sum_{j=1}^{M}\sum_{i=1}^{N}\omega_{ji}\|x_i-\mu_j-W_jS(W_j^t(x-\mu_j))\|^2-\beta\sum_{j=1}^{M}\sum_{i=1}^{N}\omega_{ji}\ln\omega_{ji},\ \sum_{j=1}^{M}\omega_{ji}=1.$$

$1\geq\omega_{ji}\geq 0$ denotes the probability of x_i from the cluster j. It is implemented by the two iterative steps:

First, with $\mu_j^{(k)}, W_j^{(k)}$'s fixed and from $\sum_{j=1}^{M}\omega_{ji}=1$ and $\nabla_{\omega_{ji}}J=0$, we can get $\omega_{ji}=e^{-\|x_i-\mu_j-W_jS(W_j^t(x-\mu_j))\|^2/\beta}/\sum_{j=1}^{M}e^{-\|x_i-\mu_j-W_jS(W_j^t(x-\mu_j))\|^2/\beta}$. Second, with the ω_{ji}'s fixed we update μ_j, W_j by one gradient ascent step on $J_j=-\sum_{i=1}^{N}\omega_{ji}\|x_i-\mu_j-W_jS(W_j^t(x-\mu_j))\|^2$.

References

[1] R.W. Brockett. *Linear Algebra Appl.*, vol. 146. 1991, pp. 79-91. [2] K.Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 1972 [3] J. Karhunen & J. Joutsensalo, Tech. Rep A17, Helsinki U. of Tech., Comp. & Infom. Sci. Lab., 1993. [4] N.Kambhatla & T.K.Leen, in *Advances in NIPS 6*, Morgan Kaufmann, San Mateo, 1994, pp 152-159. [5] E. Oja. *J. of Mathematical Biology*, vol. 16, 1982, pp. 267-273. [6] E. Oja and J. Karhunen. *J. of Math. Analysis and Applications*, vol. 106, 1985, pp. 69-84. [7] E. Oja, H. Ogawa, & J. Wangviwattana, Proc. ICANN'91, Espoo, Finland, June 1991, pp. 385-390. [8] E. Oja, H. Ogawa, and J. Wangviwattana. *IEICE Trans. on Information and Systems (Japan)*, vol. E75-D, 1992, pp. 366 - 375 (part I) and pp. 376-382 (Part II). [9] A.Sudjianto & M.H.Hassoun, *Prof. of 1994 IEEE ICNN*, Orlando, FL, Vol.II, pp1247-1252. [10] J.G. Taylor and S.G. Coombes. *Neural Networks*, vol. 6, 1993, pp. 423-427. [11] L. Xu. In *Proc. IJCNN-Singapore-91*, Nov. 1991, pp. 2362-2367 (part I), pp. 2368-2373 (part II). [12] L. Xu, E. Oja, and C.Y. Suen, *Neural Networks*, vol. 5, 1992, pp. 441-457. [13] L. Xu and A. Yuille, In *Proc. IJCNN-92-Baltimore*, Maryland, June 1992, pp. I-812-817, also in *Advances in NIPS 5*. Morgan Kaufmann, San Mateo, 1993, pp. 467-474. [14] L. Xu, *Neural Networks*, vol. 6, 1993, pp. 627-648. [15] L. Xu & M.I.Jordan, *Proc. of WCNN'93*, Portland, OR, Vol. II, 431-434. [16] L. Xu & M.I.Jordan, MIT Computational Cognitive Science, Tech. Rep. 9302, March, 1993. [17] L. Xu, *Prof. of 1994 IEEE ICNN*, Orlando, FL, Vol.II, pp1252-1257. [18] L. Xu, *Prof. of 1994 IEEE ICNN*, Orlando, FL, Vol.I, pp315-320.