# Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach: (III) Models and Algorithms for Dependence Reduction, Data Dimension Reduction, ICA and Supervised Learning *

Lei Xu

Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, NT, Hong Kong, China
Fax 852 2603 5024, Email lxu@cs.cuhk.hk, http://www.cse.cuhk.edu.hk/~lxu/

*Invited paper, in K.M. Wong, I. King and D.Y. Yeung eds,* **Theoretical Aspects of Neural Computation** : A Multidisciplinary Perspective (Hong Kong International Workshop TANC'97), Springer, pp43-60.

**Abstract.** This paper is a sister paper of [1] published in this same proceeding, for further interpreting the *Bayesian Ying-Yang (BYY) learning system and theory* through its uses on developing models and algorithms for dependence reduction, independent component analysis, data dimension reduction, supervised classification and regression with three-layer net, mixtures-of-experts, and radial basis function nets. Readers are referred to [14, 1] for the details on BYY learning system and theory. In addition, the relation of BYY learning system and theory to a number of existing learning model and theories has been discussed in [14].

## 1 BKYY Dependence Reduction System and Theory

In many applications, we want to implement a unsupervised mapping from observation $x$ into $y = [y^{(1)}, \cdots, y^{(k)}]^T$ such that the dependence among the components of $y$ is reduced as much as possible. This aim is also regarded as a basic principle in a brain perception system formed via unsupervised learning [3]. This process is studied in the literature under several names such as *Dependence Reduction*, *Factorial Learning*, *Independent Component Analysis (ICA)* , *Factorial*

* Supported by HK RGC Earmarked Grants CUHK250/94E and CUHK 339/96E and by Ho Sin-Hang Education Endowment Fund for Project HSH 95/02. The basic ideas of the BYY learning system and theory in my previous papers started three years ago— the first year of my returning to HK. As HK in transition to China, this work was in transition to its current shape. The preliminary version of this paper and its sister papers [14, 1] are all completed in the first month that HK returned to China and thus I formally returned to my motherland as well. I would like to use my this work, an effort on the harmony of an ancient Chinese philosophy and the modern western science, as a memory of this historic event.

*Encoding*. These names are very closely related, although their detailed meanings are slight different. We consider that *Dependence Reduction* may be a more general name because it covers the meanings of the other three names and also maybe more appropriate to those efforts that attempt to reduce dependence but finally may or may not really reach independence. Due to limited space, we omit to mention a quite large volume publications related to these topics.

In [14], it has been shown that we can apply the BYY learning system and theory at the special case of $y = [y^{(1)}, \cdots, y^{(k)}]^T$ with $y^{(j)}$ being binary to obtain a so called BYY *Factorial Encoding (FE)* system and theory. Here, we consider the general case that $y = [y^{(1)}, \cdots, y^{(k)}]^T$ is either binary or real such that not only the previously proposed BYY FE system and theory has been further refined and improved, but also a general *Bayesian Ying-Yang Dependence Reduction (BYY-DR)* system and theory is suggested.

Generally speaking, we can design $p_{M_y}(y)$ to be an independent parametric density or free density:

$$p_{M_y}(y) = \begin{cases} p(y|\theta_k) = \prod_{j=1}^{k} p(y^{(j)}|\xi_j), \ p(y^{(j)}|\xi_j) = \sum_{r=1}^{k_{y,j}} \alpha_{r,j} p(y^{(j)}|\xi_{r,j}), \\ p(y) = \prod_{j=1}^{k} p(y^{(j)}), \quad p(y^{(j)}) \ is \ free. \end{cases} \tag{1}$$

where $\alpha_{r,j} > 0, \sum_{r=1}^{k_{y,j}} \alpha_{r,j} = 1$, $\theta_k = \{\xi_j\}_{j=1}^{k}$. We put this $p_{M_y}(y)$ in[2] eq.[1.4], eq.[1.5] and eq.[1.6] to get BYY-DR, BCYY-DR, BKYY-DR system and theory as the special cases of BYY, BCYY, BKYY learning system and theory, respectively. In this paper, we concentrate on the BKYY-DR system and theory only. Generally speaking, it is just a direct application of using the above eq.(1) in eq.[1.6] with the general implementation technique given in Sec.3 in [1].

From Sec.3 in [1], we further get the detailed algorithms and criteria in Tab.1 with the following three architectures:

$$\begin{array}{rl} Forward: & p_{M_{x|y}}(x|y) = p(x|y) \ free, \quad p_{M_{y|x}}(y|x) = p(y|x, \theta_{y|x}), \\ Backward: & p_{M_{y|x}}(y|x) = p(y|x) \ free, \quad p_{M_{x|y}}(x|y) = p(x|y, \theta_{x|y}), \\ Bi-direction: & p_{M_{y|x}}(y|x) = p(y|x, \theta_{y|x}), \quad p_{M_{x|y}}(x|y) = p(x|y, \theta_{x|y}), \end{array} \tag{2}$$

where a density is implemented directly by a physical computing device if it is a parametric model, or indirectly by other components if it is free.

In Tab.1, Part A is obtained from eq.[1.6] and eq.[1.17] for the general implementation. Part B provides the relation to some existing methods and will be further discussed in the next section. The adaptive algorithm given in Part C is obtained according to the general implementation technique given in Sec.3 in [1], with $p_r(y) = p_{M_y}(y)$. The model selection criteria are given in Part D.

## 2 BKYY DR for ICA and Blind Source Separation

We consider the following *noisy* post-nonlinear instantaneous mixture that the $d$ dimensional observation $x$ comes from $k$ independent sources $s^{(1)}, \cdots, s^{(k)}$ via

$$x = g(Ay) + e, \ Ey = 0, \ Ee = 0, \ Eye^T = 0, \ g(Ay) = [g_1(\hat{x}^1), \cdots, g_d(\hat{x}^d)], \ \hat{x} = Ay \tag{3}$$

[2] In this paper, we will frequently refer to the equations in paper [1], which is published in this volume too. For convenience, we simply use eq.[1.n] to denote eq.(n) in [1].

**Table 1. BKYY-DR System and Theory**

| | Forward | Bi-direction | Backward |
|---|---|---|---|
| | For $KL(M_1, M_2)$ given in eq.[1.6] , $p_{M_y}(y)$ given by eq.(1) $p_{M_x}(x)$ is fixed at a nonparametric estimation based on $D_x = \{x_i\}_{i=1}^N$ | | |
| | **Part A: The Alternative Minimization Procedure** | | |
| Step 1 | Fix $p_{M_{x\|y}}(x\|y), p_{M_y}(y)$, update | | |
| | the parameter $\theta_{y\|x}$ of $p_{M_{y\|x}}(y\|x)$ to reduce $Kl(M_1, M_2)$ in eq.[1.11] e.g., moving one step along the gradient descent direction | | $p_{M_{y\|x}}(y\|x) = \frac{p_{M_{x\|y}}(x\|y)p_{M_y}(y)}{p_{M_{x\|y,y}}(x)}$ $p_{M_{x\|y,y}}(x) = \int p_{M_{x\|y}}(x\|y)p_{M_y}(y)dy$ |
| Step 2 | Fix $p_{M_{y\|x}}(y\|x)$, update | | |
| | $p_{M_{x\|y}}(x\|y) = \frac{p_{M_{y\|x}}(y\|x)p_{M_x}(x)}{p_{M_{y\|x,x}}(y)}$ $p_{M_{y\|x,x}}(y) = \int p_{M_{y\|x}}(y\|x)p_{M_x}(x)dx$ | | the parameter $\theta_{x\|y}$ of $p_{M_{x\|y}}(x\|y)$ to increase $\int_{x,y} p_{M_{y\|x}}(y\|x)p_{M_x}(x)\ln p_{M_{x\|y}}(x\|y)dxdy$ e.g., moving one step along the gradient ascent direction |
| | For $p_{M_y}(y)$ free, let $p_{M_y}(y_j) = \int p_{M_{y\|x}}(y_j\|x)p_{M_x}(x)dx$, $j = 1, \cdots, k$ For $p_{M_y}(y)$ parametric, update its parameter to increase $\int_{x,y} p_{M_{y\|x}}(y\|x)p_{M_x}(x)\ln p_{M_y}(y)dxdy$ e.g., moving one step along the gradient ascent direction . | | |
| | **Part B: $\min_{M_1,M_2} KL(M_1, M_2)$ is equivalent to minimize** | | |
| | $\int_y p_{M_{y\|x,x}}(y)\ln\frac{p_{M_{y\|x,x}}(y)}{p_{M_y}(y)}dy$ with $p_{M_{y\|x,x}}(y)$ and $p_{M_{x\|y}}(x\|y)$ given above | | $\int_x p_{M_x}(x)\ln\frac{p_{M_x}(x)}{p_{M_{x\|y,y}}(x)}dx$ with $p_{M_{x\|y,y}}(x)$ and $p_{M_{y\|x}}(y\|x)$ given above |
| | **Part C: The Stochastic Approximation Adaptive Algorithm** | | |
| Step 1 | Refer to Sec.3 in [1], use $p_{M_y}(y)$ by eq.(1) as the *Sampling Reference* density, get a sample $x_t$ and randomly take a sample $y_t$ from $p_{M_y}(y)$. Fix $p_{M_{x\|y}}(x\|y), p_{M_y}(y)$, update | | |
| | $\theta_{y\|x}^{new} = \theta_{y\|x}^{old} - \frac{\eta}{p_{M_y}(y_t)} \times \frac{\partial[p_{M_{y\|x}}(y_t\|x_t)\ln\frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_{x\|y}}(x_t\|y_t)p_{M_y}(y_t)}]}{\partial\theta_{y\|x}}\|_{\theta_{y\|x}=\theta_{y\|x}^{old}}$ | | Get $p_{M_{y\|x}}(y\|x)$ the same as in Part A |
| Step 2 | Fix $p_{M_{y\|x}}(y\|x)$, update | | |
| | Get $p_{M_{x\|y}}(x\|y)$ above (in the batch way only) | | $\theta_{x\|y}^{new} = \theta_{x\|y}^{old} + \frac{\eta}{p_{M_y}(y_t)} \times \frac{\partial p_{M_{y\|x}}(y_t\|x_t)\ln p_{M_{x\|y}}(x_t\|y_t)}{\partial\theta_{x\|y}}\|_{\theta_{x\|y}=\theta_{x\|y}^{old}}$ |
| | When $p_{M_y}(y)$ free, the same as in Part A(in the batch way only ). For $p_{M_y}(y)$ parametric, $\theta_y^{new} = \theta_y^{old} + \frac{\eta}{p_{M_y}(y_t)}\frac{\partial p_{M_{y\|x}}(y_t\|x_t)\ln p_{M_y}(y_t)}{\partial\theta_y}\|_{\theta_y=\theta_y^{old}}$ | | |
| | **Part D: The Model Selection Criteria for $k$** | | |
| | After learning, denote the results by $p^*_{M_{y\|x}}(y\|x), p^*_{M_{x\|y}}(x\|y), p^*_{M_y}(y)$ $D_x = \{x_t\}_{t=1}^N$ and randomly take samples $D_y = \{y_\tau\}_1^{N'}$ from $p^*_{M_y}(y)$ From eqs.[1.13&14], get $k^* = arg\min_k J_1(k)$ or $k^* = arg\min_k J_2(k)$ $J_1(k) = \frac{1}{NN'}\sum_{t=1}^N\sum_{\tau=1}^{N'}\frac{p^*_{M_{y\|x}}(y_\tau\|x_t)}{p^*_{M_y}(y_\tau)}\ln\frac{p^*_{M_{y\|x}}(y_\tau\|x_t)}{p^*_{M_{x\|y}}(x_t\|y_\tau)p^*_{M_y}(y_\tau)}$ $J_2(k) = -\frac{1}{NN'}\sum_{t=1}^N\sum_{\tau=1}^{N'}\frac{p^*_{M_{y\|x}}(y_\tau\|x_t)}{p^*_{M_y}(y_\tau)}\ln[p^*_{M_{x\|y}}(x_t\|y_\tau)p^*_{M_y}(y_\tau)]$. | | |

**Table 2. BKYY DR for ICA and Blind Source Separation (BSS)**

| Forward | Bi-direction | Backward |
|---|---|---|
| $Kl(M_1, M_2)$ and $p_{M_x}(x)$ are the same as in Tab.1<br>$Kl(M_1, M_2) = \int p_{M_x}(x)(H(y\|x) - Q(x\|y) - C(y))dxdy$, $H(y\|x) = p_{M_{y\|x}}(y\|x) \ln p_{M_{y\|x}}(y\|x)$<br>$Q(x\|y) = p_{M_{y\|x}}(y\|x) \ln p_{M_{x\|y}}(x\|y)$, $C(y) = p_{M_{y\|x}}(y\|x) \ln p_{M_y}(y)$ | | |
| **Part A: Architecture Design** | | |
| For binary $y$, $p_{M_y}(y) = \prod_{j=1}^{k} q_j^{y^{(j)}}(1-q_j)^{1-y^{(j)}}$, $\theta_k = \{q_j : 0 \le q_j \le 1\}_{j=1}^{k}$<br>For real $y$, $p_{M_y}(y) = p(y\|\theta_k)$ given by eq.(1) | | |
| $p_{M_{x\|y}}(x\|y)$ is free | $p_{M_{x\|y}}(x\|y) = G(x, g(Ay), \Sigma)$ | |
| For binary $y$, $p_{M_{y\|x}}(y\|x) = \prod_{j=1}^{k} \mu_j^{y_j}(1-\mu_j)^{1-y_j}$<br>$\mu_j = s(e_j^T[Wf(x,\theta_f) + By])$, $H(y\|x) =$<br>$p_{M_{y\|x}}(y\|x) \sum_{j=1}^{k}[y_j \ln \mu_j + (1-y_j)\ln(1-\mu_j)]$<br>For real $y$, $p_{M_{y\|x}}(y\|x) = \sum_{j=1}^{n_{y\|x}} \beta_j G(y - Wf(x,\theta_f), 0, \Pi_j)$ | | $p_{M_{y\|x}}(y\|x)$ is free |
| Note: 1. $s(r)$ is sigmoid with its range on $[0,1]$, e.g., $s(r) = 1/(1+e^{-r})$<br>2. $e_j^T x$ gives the $j$-th element of the vector $x$.<br>3. The nonlinear function $f(x,\theta_f)$ can be implemented by a forward network. | | |
| **Part B: Adaptive Algorithm** | | |
| Step 1: the same as Part C in Tab.1, get a sample $x_t$ and randomly take a sample $y_t$ from $p_{M_y}(y)$. Fix $p_{M_{x\|y}}(x\|y), p_{M_y}(y)$, update | | |
| $\theta_{y\|x} = \{W, B, \theta_f, \{\Pi_j\}\}$ or $\theta_{y\|x} = \{W, B, \theta_f\}$,<br>$\theta_{y\|x}^{new} = \theta_{y\|x}^{old} - \frac{\eta}{p_{M_y}(y_t)} \frac{\partial H(y_t\|x_t) - Q(x_t\|y_t) - C(y_t)}{\partial \theta_{y\|x}}\|_{\theta_{y\|x} = \theta_{y\|x}^{old}}$ | | $p_{M_{y\|x}}(y\|x) = \frac{p_{M_{x\|y}}(x\|y)p_{M_y}(y)}{\int p_{M_{x\|y}}(x\|y)p_{M_y}(y)dy}$ |
| Step 2: | Fix $p_{M_{y\|x}}(y_t\|x_t)$, update | |
| (in batch way only)<br>$p_{M_{x\|y}}(x\|y) = \frac{p_{M_{y\|x}}(y\|x)p_{M_x}(x)}{\int p_{M_{y\|x}}(y\|x)p_{M_x}(x)dx}$ | $g(\hat{x}, \phi) = [g_1(\hat{x}^1, \phi), \cdots, g_d(\hat{x}^d, \phi)]$, $\hat{x} = Ay$,<br>$G_t = \frac{\partial g^T(\hat{x},\phi)}{\partial \hat{x}}\|_{\hat{x}=A^{old}y_t, \phi=\phi^{old}}(\Sigma^{old})^{-1}$<br>$A^{new} = A^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)} G_t[x_t - g(A^{old}y_t, \phi^{old})]y_t^T$,<br>$H_t = \frac{\partial g^T(\hat{x},\phi)}{\partial \phi}\|_{\hat{x}=A^{old}y_t, \phi=\phi^{old}}(\Sigma^{old})^{-1}$,<br>$\phi^{new} = \phi^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)} H_t[x_t - g(A^{old}y_t, \phi^{old})]$;<br>Get $a(x_t, y_t) = x_t - g(A^{new}y_t, \phi^{new})$,<br>$\Sigma^{new} = (1-\eta)\Sigma^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)} a(x_t, y_t)a^T(x_t, y_t)$. | |
| | Particularly, for linear $g(u) = u$, we have<br>$A^{new} = A^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)}(x_t - A^{old}y_t)y_t^T$, $\Sigma^{new} =$<br>$(1-\eta)\Sigma^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)}(x_t - A^{new}y_t)(x_t - A^{new}y_t)^T$<br>$W^{new} = W^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)}(y_t - W^{old}x_t)x_t^T$ ( backward case). | |
| For real $y$ and $p_{M_y}(y)$ is parametric<br>$\xi_j^{new} = \xi_j^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)} \frac{\partial \ln p(y_t^{(j)}\|\xi_j)}{\partial \xi_j}\|_{\xi_j = \xi_j^{old}}$, $j = 1, \cdots, k$,<br>For binary $y$, $q_j^{new} = \frac{1}{1+exp(-r_j^{new})}$, $r_j^{new} = r_j^{old} + \eta \frac{p_{M_{y\|x}}(y_t\|x_t)}{p_{M_y}(y_t)}(y_\tau^{(j)} - q_j^{old})$. | | |
| **Part C: Three Ways of Recovery $x \to y$** | | |
| (1). Random sampling $\hat{y}$ according to the resulted $p_{M_{y\|x}}(y\|x)$.<br>(2). Maximum posterior decision. $\hat{y} = arg \max_y p_{M_{y\|x}}(y\|x)$.<br>(3) For linear $g(u) = u$, get a direct inverse mapping $\hat{y} = Wx$ | | |
| **Part D: The Criteria for Selecting $k$ (the same as in Tab.1)** | | |

**Table 3. Adaptive Algorithm with Finite-Mixture for Noiseless ICA**

| Gaussian Mixture | Derivative Sigmoid Mixture |
|---|---|
| With $\theta_k^{old}$ fixed, update $W^{new} = W^{old} + \eta(I + \phi(y)y^T)W^{old}$, $\phi(y) = [\phi_1(y_1), \cdots, \phi_k(y_k)]^T$ | |
| $y = W^{old}x$, $y^{(j)} = x_t^T w_j^{old}$, $\phi_j(y^{(j)}) = \frac{\partial \ln p(y^{(j)}\|\xi_j)}{\partial y^{(j)}} = \frac{\sum_{r=1}^{k_{y,j}} \alpha_{r,j}^{old} \partial p(y^{(j)}\|\xi_{r,j}^{old})/\partial y^{(j)}}{\sum_{r=1}^{k_{y,j}} \alpha_{r,j}^{old} p(y^{(j)}\|\xi_{r,j}^{old})})$ | |
| $p(y^{(j)}\|\xi_{r,j}) = G(y^{(j)}, \mu_{r,j}, \sigma_{r,j}^2)$, $\xi_{r,j} = \{\mu_{r,j}, \sigma_{r,j}^2\}$, $\frac{\partial p(y^{(j)}\|\xi_{r,j})}{\partial y^{(j)}} = -\frac{(y^{(j)} - \mu_{r,j})}{\sigma_{r,j}^2} p(y^{(j)}\|\xi_{r,j})$ | $p(y^{(j)}\|\xi_{r,j}) = \frac{ds(y^{(j)}\|\xi_{r,j})}{dy^{(j)}} = \frac{b_{r,j}e^{-b_{r,j}(y^{(j)} - a_{r,j})}}{(1+e^{-b_{r,j}(y^{(j)} - a_{r,j})})^2}$, $s(y^{(j)}\|\xi_{r,j}) = \frac{1}{1+e^{-b_{r,j}(y^{(j)} - a_{r,j})}}$, $\xi_{r,j} = \{a_{r,j}, b_{r,j}\}$, $\frac{\partial p(y^{(j)}\|\xi_{r,j})}{\partial y^{(j)}} = \frac{b_{r,j}^2 e^{-b_{r,j}(y^{(j)} - a_{r,j})}(e^{-b_{r,j}(y^{(j)} - a_{r,j})} - 1)}{(e^{-b_{r,j}(y^{(j)} - a_{r,j})} + 1)^3}$ |
| Step 1: $y^{(j)} = x_t^T w_j^{new}$, $h_{r,j}(x) = \frac{\alpha_{r,j}^{old} p(y^{(j)}\|\xi_{r,j}^{old})}{\sum_{r=1}^{k_{y,j}} \alpha_{r,j}^{old} p(y^{(j)}\|\xi_{r,j}^{old})}$, $\alpha_{r,j}^{new} = (1-\gamma)\alpha_{r,j}^{old} + \gamma h_{r,j}(x)$ | |
| $\xi_{r,j}^{new} = \xi_{r,j}^{old} + \gamma \frac{h_{r,j}(x)}{\alpha_{r,j}^{new}} \frac{\partial \ln p(y^{(j)}\|\xi_{r,j}^{old})}{\partial \xi_{r,j}}$, it takes the specific form as follows, respectively | |
| Step 2: $\mu_{r,j}^{new} = (1-\gamma)\mu_{r,j}^{old} +$ $\gamma \frac{h_{r,j}(x)}{\alpha_{r,j}^{new}} y^{(j)}$, $(\sigma_{r,j}^2)^{new} = (1-\gamma)\times$ $(\sigma_{r,j}^2)^{old} + \gamma \frac{h_{r,j}(x)}{\alpha_{r,j}^{new}}[y^{(j)} - \mu_{r,j}^{new}]^2$ | Step 2: $b_{r,j}^{new} = b_{r,j}^{old} +$ $\gamma \frac{h_{r,j}(x)}{\alpha_{r,j}^{new}}[\frac{1}{b_{r,j}^{old}} - (y^{(j)} - a_{r,j}^{new})\frac{1-e^{-b_{r,j}^{old}(y^{(j)} - a_{r,j}^{new})}}{1+e^{-b_{r,j}^{old}(y^{(j)} - a_{r,j}^{new})}}]$ $a_{r,j}^{new} = a_{r,j}^{old} + \gamma \frac{h_{r,j}(x)}{\alpha_{r,j}^{new}} b_{r,j}^{old} \frac{1-e^{-b_{r,j}^{old}(y^{(j)} - a_{r,j}^{old})}}{1+e^{-b_{r,j}^{old}(y^{(j)} - a_{r,j}^{old})}}$ |
| Then, let $W^{old} = W^{new}$, $\theta_k^{old} = \theta_k^{new}$. | |
| We can even simply let $\sigma_{r,j}^2 = 1, b_{r,j} = 1$, $k_{y,j} = 2$ and $\alpha_{r,j} = 0.5$ such that the algorithms can be simplified considerably by removing the updating on $\sigma_{r,j}^2, b_{r,j}, \alpha_{r,j}$. | |

with $x$ being strict wide stationary and ergodic and $e$ being noise. The purpose of *Blind Source Separation (BSS)* is to get $\hat{y}$ which recovers $y$ up to only constant unknown scales and any permutation of indices. In the case that $e = 0$, $d = k$ and $g(u) = u$ linear, it reduces to the well known linear instantaneous mixture, which is solved when $\hat{y} = Wx$ makes the components of $\hat{y}$ becomes independent. Thus, it is also called *Independent Component Analysis (ICA)*, studied widely in the literature. Due to limited space, we omit to mention one by one all the existing references and they can be found in a very recent overview paper [8]. In the case that $e = 0$, $d = k$ and $g(u)$ nonlinear, eq.(3) reduces to the post-nonlinear instantaneous mixture studied recently by [13]. In this paper, we consider the general case eq.(3) with $e \neq 0$, $d \neq k$, nonlinear $g(u)$ and unknown $k$, by directly using the BYY DR introduced in Sec.1.

When $e$ is gaussian $G(e, 0, \Sigma)$, eq.(3) can be described by $p_{M_{x|y}}(x|y) = G(x, g(Ay), \Sigma)$. Putting this specific setting in Tab.1 and together with appropriate designs as given in Part A of Tab.2, we can get adaptive algorithm for the BSS problem eq.(3) directly, as given by Part B in Tab.2, where the batch way ICA algrotihm is also given for the forward case. Moreover, Part C in Tab.2 suggests three ways of recovering $y$ from $x$. The first two are directly understandable. In the backward case, there is originally no $W$ for the third way. However we can indirectly get one via $\min_W J_2(W)$, $J_2(W) = E\|y - \hat{y}\|^2 = Tr[E(y - Wx)(y - Wx)^T]$. This $W$ can also be adaptively learned together with the adaptation on $A$, as shown by the last line in the middle block of Part B. Finally, the criteria for

selecting the unknown number of sources keep the same as Part D in Tab.1.

It is interesting to consider the noiseless special case $e = 0$. When $k$ is smaller than its correct value, we have effectively $|\Sigma| \neq 0$ during the learning, and thus the situation is similar to the case with noise. When $k$ becomes equal or larger than its correct value, $\Sigma$ will become singular as other parameters converge to the correct values. In other words, we can still use the adaptive learning algorithm in Tab.2 for the backward and Bi-direction cases directly. $\Sigma$ becoming singular is just a signal that indicates the correct convergence. That is, we can start at a small value for $k$ and then gradually increase it, and then stop the learning once $\Sigma$ becomes singular.

For the forward case with linear $g(u) = u$ and $e = 0$, we have $p_{M_{y|x}}(y|x) = \delta(x - W^{-1}y)/|W|$, $p_{M_{y|x,x}}(y) = \int_x p_{M_{y|x}}(y|x)p_{M_x}(x)dx = p_{M_x}(W^{-1}y)/|W| = p(y)$, thus from Part B in Tab.1, $\min_{M_1,M_2} KL(M_1, M_2)$ is equivalent to minimize $\int_y p(y) \ln [p(y)/\prod_{j=1}^k p_{M_y}(y^{(j)})]dy$, $y = Wx$. If we further let $p_{M_y}(y^{(j)}) = p(y^{(j)})$, it reduces exactly to the minimum mutual information (MMI) criterion used by [2] and also equivalent to maximum likelihood ICA [6, 12], INFORMAX [4]. The MMI criterion is usually implemented by gradient algorithm. An improved adaptive natural gradient algorithm is used in [2]:

$$W^{new} = W^{old} + \eta \Delta W, \quad \Delta W = (I + \phi(y)y^T)W,$$
$$\phi(y) = [\phi_1(y^{(1)}), \cdots, \phi_k(y^{(k)})]^T, \quad \phi_j(y^{(j)}) = \frac{\partial ln p(y^{(j)}|\xi_j)}{\partial y^{(j)}}. \tag{4}$$

These mentioned efforts have reached certain successes for sources of either only super-gaussians or only sub-gaussians [16]. The key point is the difference in the use of the parameter form for $p(y^{(j)}|\xi_j)$.

From Tab.2, we can get a new adaptive noiseless ICA algorithm, based on eq.(4) and eq.(1). Its specific versions for $p(y^{(j)}|\xi_{r,j})$ being either gaussian or defined by sigmoid function are given in Tab.3. This use of finite mixture density for $p(y^{(j)}|\xi_{r,j})$ makes the algorithm become more flexible to adapt different sources. Experiments have shown that it works well for various kinds of sources, including those on which those mentioned efforts succeeded and failed [16].

## 3 BKYY Data Dimension Reduction (DDR)

In Sec.9 of [14], it has been shown that we can also use the Basis BYY learning theory eqs.[1.4][1.5][1.6] as a general data dimension reduction (DDR) system and theory with the following design:

$$p_{M_y}(y) = \sum_{j=1}^{n_y} \alpha_j p(y|\xi_j), \; p_{M_{x|y}}(x|y) = \sum_{j=1}^{n_{x|y}} \gamma_j p(x - f(y, A_j)|\phi_j),$$
$$p_{M_{y|x}}(y|x) = \sum_{j=1}^{n_{y|x}} \beta_j p(y|x, g(x, W_j), \psi_j), \tag{5}$$

for solving the problem of mapping the observed high dimension data $x$, generated from a unknown $y = [y_1, \cdots, y_k] \in R^k$ under noise, back to its original $R^k$.

**Table 4. BKYY Learning for Three Layer Forward Nets ( General Case)**

**1. Variable Types**

$y = [y_1, \cdots, y_k]$ with either $y_j \in R$ or $y_j \in \{0, 1\}$,
Binary-F: $z = [z_1, \cdots, z_m]$ with either $z_j \in R$ or $z_j \in \{0, 1\}$, without constraint.
Binary-E: $z$ of Binary-F plus the exclusive constraint $\sum_{j=1}^{m} z_j = 1$.

**2. Architecture Design**

$p_{M_{z|y}}(z|y) = p(z|y, \theta_{z|y}) =:$

$$\begin{cases} \prod_{j=1}^{m} \pi_j^{z_j}(1-\pi_j)^{1-z_j}, & \text{Binary-F } z, \\ \sum_{j=1}^{m} z_j \pi_j / \sum_{j=1}^{m} \pi_j, & \text{Binary-E } z, \\ G(z, g(y, W_{z|y}), \Sigma_{z|y}), & \text{Real } z. \end{cases}$$

$\pi_j = g_j(y, W_{z|y})$,
$g(y, W_{z|y}) = [g_1(y, W_{z|y}), \cdots, g_m(y, W_{z|y})]$
nonlinear functions in general
e.g., $g_j(y, W_{z|y}) = s(e_j^T(W_{z|y} y))$

$p_{M_{y|x}}(y|x) = p(y|x, \theta_{y|x}) =:$

$$\begin{cases} \prod_{j=1}^{k} \mu_j^{y_j}(1-\mu_j)^{1-y_j}, & \text{binary } y, \\ G(y, f(x, W_{y|x}), I), & \text{Real } y. \end{cases}$$

$\mu_j = f_j(x, W_{y|x})$
$f(x, W_{y|x}) = [f_1(x, W_{y|x}), \cdots, f_k(x, W_{y|x})]$
nonlinear functions in general
e.g., $f_j(x, W_{y|x}) = s(e_j^T(W_{y|x} x))$

Free $p_{M_{y|x,z}}(y|x, z) = p(y|x, z)$:
$p(y|x, z) = \frac{p(z|y, \theta_{z|y}) p(y|x, \theta_{y|x})}{p_{M_2}(z|x)}$, $p_{M_2}(z|x) = \int_y p(z|y, \theta_{z|y}) p(y|x, \theta_{y|x}) dy$

Parametric $p_{M_{y|x,z}}(y|x, z) = p(y|x, z, \theta_{y|x,z})$:
For binary $y$, $\prod_{j=1}^{k} \kappa_j^{y_j}(1-\kappa_j)^{1-y_j}$, $\kappa_j = s(h_j(x, y, W_{y|x,z}))$
$h(x, y, W_{y|x,z}) = [h_1(x, y, W_{y|x,z}), \cdots, h_k(x, y, W_{y|x,z})]$,
e.g., $h_j(x, y, W_{y|x,z}) = e_j^T(W_{y|x,z}[x^T, z^T]^T)$
For real $y$, $p(y|x, z, \theta_{y|x,z}) = G(y, h(x, y, W_{y|x,z}), I)$

Note: 1. $s(r)$ is sigmoid with its range on $[0, 1]$, e.g., $s(r) = 1/(1+e^{-r})$
For Binary-E $z$, $s(r)$ may be monotonic increasing , e.g., $s(r) = e^r$
2. $e_j^T x$ gives the $j$-th element of the vector $x$.
3. the equation for specifying the free $p(y|x, z)$ is given by Theorem 1 in [1]

**3. Learning, i.e.,** $\min_{\{\theta_{y|x,z}, \theta_{z|y}, \theta_{y|x}, k\}} J(\theta_{y|x,z}, \theta_{z|y}, \theta_{y|x}, k)$

$J(\theta_{y|x,z}, \theta_{z|y}, \theta_{y|x}, k) = -H_{y|x,z} - L_{z|y} - L_{y|x}$
$H_{y|x,z} = \int_{x,z} H_{y|x,z}(x, z) p_{M_1^L}(z, x) dx dz$,
$H_{y|x,z}(x, z) = -\int_y p_{M_{y|x,z}}(y|x, z) \ln p_{M_{y|x,z}}(y|x, z) dy$
$L_{z|y} = \int_{x,z} L_{z|y}(x, z) p_{M_1^L}(z, x) dx dz$, $L_{z|y}(x, z) = \int_y p_{M_{y|x,z}}(y|x, z) \ln p(z|y, \theta_{z|y}) dy$
$L_{y|x} = \int_{x,z} L_{y|x}(x, z) p_{M_1^L}(z, x) dx dz$, $L_{y|x}(x, z) = \int_y p_{M_{y|x,z}}(y|x, z) \ln p(y|x, \theta_{y|x}) dy$
For free $p_{M_{y|x,z}}(y|x, z) = p(y|x, z)$,
$J(\theta_{z|y}, \theta_{y|x}, k) = -L_{z|x}$, $L_{z|x} = \int_{x,z} p_{M_1^L}(z, x) \ln p_{M_2}(z|x) dx dz$
When $p_{M_1^L}(z, x) = p_{M_x}(x) p_{M_{z|x}}(z|x)$ given by eq.[1.2] and eq.[1.8],
$H_{y|x,z} = \sum_{(x,z) \in D_{x,z}} H_{y|x,z}(x, z)$, $L_{z|y} = \sum_{(x,z) \in D_{x,z}} L_{z|y}(x, z)$
$L_{z|x} = \sum_{(x,z) \in D_{x,z}} \ln p_{M_2}(z|x)$, $L_{y|x} = \sum_{(x,z) \in D_{x,z}} L_{y|x}(x, z)$

**4. Parameter Learning Algorithm at a fixed $k$**

Step 1: Fix $\theta_{z|y}, \theta_{y|x}$, either let the free $p(y|x, z)$ given by Part 2 above
or update $\theta_{y|x,z}$ by $\min_{\theta_{y|x,z}} J(\theta_{y|x,z}, \theta_{z|y}, \theta_{y|x}, k)$
e.g., moving one step along the gradient descent direction.
Step 2: Fix $p_{M_{y|x,z}}(y|x, z)$, update $\theta_{z|y}$ by $\max_{\theta_{z|y}} L_{z|y}$ and $\theta_{y|x}$ by $\max_{\theta_{y|x}} L_{y|x}$,
e.g., moving one step along the gradient ascent direction.

**5. Model Selection** (*selecting the number $k$ of hidden unit*)

With $\theta^*_{y|x,z}, \theta^*_{z|y}, \theta^*_{y|x}$ obtained by the above algorithm, select $k^*$ by
From eqs.[1.13&14], by $J_1(k) = J(\theta^*_{y|x,z}, \theta^*_{z|y}, \theta^*_{y|x}, k)$
or $J_2(k) = -(L_{z|y} + L_{y|x})|_{\{\theta^*_{y|x,z}, \theta^*_{z|y}, \theta^*_{y|x}\}}$

**Table 5. BKYY Learning for Three Layer Forward Nets ( Special Cases)**

**1. $\delta$-Yang based system**

$p_{M_{y|x,z}}(y|x,z) = p(y|x,z)$ free, $p(y|x,\theta_{y|x}) = \delta(y - f(x,W_{y|x}))$,
i.e., $x \to y$ by a deterministic mapping $y = f(x,W_{y|x})$.
From Part 3 in Tab.4, $\min_{\{\theta_{z|y},\theta_{y|x}\}} J(\theta_{z|y},\theta_{y|x},k)$ becomes
ML learning : $\max_{\{\theta_{z|y},W_{y|x}\}} L_{z|x}$, $L_{z|x} = \sum_{(x,z)\in D_{x,z}} \ln p(z|f(x,W_{y|x}),\theta_{z|y})$
For $p(z|y,\theta_{z|y}) = G(z,g(y,W_2),\sigma^2 I)$, it becomes
$\min_{\{W_{z|y},W_{y|x}\}} \sum_{(x,z)\in D_{x,z}} \|z_i - g(f(x_i,W_{y|x}),W_{z|y})\|^2$

**2. Smoothed Yang based system**

$p_{M_{y|x,z}}(y|x,z) = p(y|x,z)$ free, $p(z|y,\theta_{z|y}) = p(z|E(y|x,\theta_{y|x}),\theta_{z|y})$,
$E(y|x,\theta_{y|x}) = \int_y y p(y|x,\theta_{y|x})dy = f(x,W_{y|x})$
$p_{M_2}(z|x) = p(z|f(x,W_{y|x}),\theta_{z|y})$, $H_{y|x,z}(x,z) = -L_{y|x}(x,z)$
$J(\theta_{z|y},\theta_{y|x},k) = -\sum_{(x,z)\in D_{x,z}} \ln p(z|f(x,W_{y|x}),\theta_{z|y})$
thus the parameter learning is the same as in the above $\delta$-Yang based system
However, for selecting $k$, by eqs.[1.13&14] $J(k) = J_2(\theta^*_{z|y},\theta^*_{y|x},k)$, $J_2(\theta_{z|y},\theta_{y|x},k) =$
$-\sum_{(x,z)\in D_{x,z}}[\ln p(z|f(x,W_{y|x}),\theta_{z|y}) + \int_y p(y|x,\theta_{y|x}) \ln p(y|x,\theta_{y|x})dy]$
$\int_y p(y|x,\theta_{y|x}) \ln p(y|x,\theta_{y|x})dy = -\sum_{j=1}^k [\mu_j \ln \mu_j + (1-\mu_j)\ln(1-\mu_j)]$, binary $y$

**3. Mean-Field Yang based system**

$p_{M_{y|x,z}}(y|x,z) = p(y|x,z)$ free. In all the integrals over $y$,
we let $y$ approximately replaced by $E(y|x,\theta_{y|x}) = f(x,W_{y|x})$
$p_{M_2}(z|x) = p(z|f(x,W_{y|x}),\theta_{z|y})p(f(x,W_{y|x})|x,\theta_{y|x})$
$p(y|x,z) = \frac{p(z|y,\theta_{z|y})p(y|x,\theta_{y|x})}{p(z|f(x,W_{y|x}),\theta_{z|y})p(f(x,W_{y|x})|x,\theta_{y|x})}$, $H_{y|x,z}(x,z) = 0$,
$L_{z|y}(x,z) = \ln p(z|f(x,W_{y|x}),\theta_{z|y})$, $L_{y|x}(x,z) = -\ln p(f(x,W_{y|x})|x,\theta_{y|x})$
$\ln p(f(x,W_{y|x})|x,\theta_{y|x}) = \begin{cases} 0, & \text{for real } y, \\ -\sum_{j=1}^k [\mu_j \ln \mu_j + (1-\mu_j)\ln(1-\mu_j)], & \text{for binary } y \end{cases}$
$J_1(\theta_{z|y},\theta_{y|x},k) = J_2(\theta_{z|y},\theta_{y|x},k) = J(\theta_{z|y},\theta_{y|x},k)$,
$J(\theta_{z|y},\theta_{y|x},k) = -\sum_{(x,z)\in D_{x,z}}[\ln p(z|f(x,W_{y|x}),\theta_{z|y}) + \ln p(f(x,W_{y|x})|x,\theta_{y|x})]$

**4. The Stochastic Approximation Adaptive Algorithm**

$p_{M_{y|x,z}}(y|x,z) = p(y|x,z,\theta_{y|x,z})$, and $p(z|y,\theta_{z|y})$, $p(y|x,\theta_{y|x})$ are parametric given in Tab.4
Refer to Sec.3 in [1], use $p(y|x,\theta_{y|x})$ as the *Sampling Reference* density,
Get a sample $x_t, z_t$ and randomly take a sample $y_t$ from $p(y|x_t,\theta_{y|x})$.
Step 1: Fix $\theta_{z|y}, \theta_{y|x}$, $\theta^{new}_{y|x,z} = \theta^{old}_{y|x,z} -$
$\frac{\eta}{p(y|x_t,\theta_{y|x})} \frac{\partial[p(y_t|x_t,z_t,\theta_{y|x,z}) \ln \frac{p(y_t|x_t,z_t,\theta_{y|x,z})}{p(z_t|y_t,\theta_{z|y})p(y_t|x_t,\theta_{y|x})}]}{\partial\theta_{y|x,z}}|_{\theta_{y|x,z}=\theta^{old}_{y|x,z}}$
Step 2: Fix $\theta_{y|x,z}$, update $\theta^{new}_{z|y} = \theta^{old}_{z|y} + \eta \frac{p(y_t|x_t,z_t,\theta_{y|x,z})}{p(y|x_t,\theta_{y|x})} \frac{\partial \ln p(z_t|y_t,\theta_{z|y})}{\partial\theta_{z|y}}|_{\theta_{z|y}=\theta^{old}_{z|y}}$
$\theta^{new}_{y|x} = \theta^{old}_{y|x} + \eta \frac{p(y_t|x_t,z_t,\theta_{y|x,z})}{p(y|x_t,\theta_{y|x})} \frac{\partial \ln p(y_t|x_t,\theta_{y|x})}{\partial\theta_{y|x}}|_{\theta_{y|x}=\theta^{old}_{y|x}}$

**Part 5 Three Ways of Mapping $x_t \to z_t$**

(1): $\hat{z}_t = g(f(x_t,W_{y|x}),W_{z|y})$,
(2): For binary $y$, $\hat{y}_t = arg\min_y p(y|x_t,\theta_{y|x})$ and then $\hat{z}_t = g(\hat{y}_t,W_{z|y})$,
(3): $\hat{y}_t = \int Ep(z|y,\theta_{z|y})p(y|x,\theta_{y|x})dy \approx g(f(x_t,W_{y|x}),W_{z|y})p(f(x_t,W_{y|x})|x,\theta_{y|x})$

**6. Model Selection** (*selecting the number $k$ of hidden unit*)

$D = \{x_t,z_t\}_{t=1}^N$ and randomly take samples $D_y = \{y_{\tau,t}\}_{\tau=1}^{N'}$ from $p(y|x_t,\theta^*_{y|x})$
From eqs.[1.13&14], get $k^* = arg\min_k J_1(k)$ or $k^* = arg\min_k J_2(k)$
$J_1(k) = \frac{1}{NN'}\sum_{t=1}^N \sum_{\tau=1}^{N'} \frac{1}{p(y_\tau|x_t,\theta^*_{y|x})} p(y_\tau|x_t,z_t,\theta^*_{y|x,z}) \ln \frac{p(y_\tau|x_t,z_t,\theta^*_{y|x,z})}{p(z_t|y_\tau,\theta^*_{z|y})p(y_\tau|x_t,\theta^*_{y|x})}$
$J_2(k) = -\frac{1}{NN'}\sum_{t=1}^N \sum_{\tau=1}^{N'} \frac{1}{p(y_\tau|x_t,\theta^*_{y|x})} p(y_\tau|x_t,z_t,\theta^*_{y|x,z}) \ln [p(z_t|y_\tau,\theta^*_{z|y})p(y_\tau|x_t,\theta^*_{y|x})]$

This DDR theory aims at not only modeling the generating process of data $x$ by $p_{M_{x|y}}(x|y)$ and the backward-mapping by $p_{M_{y|x}}(y|x)$, but also at discovering the original dimension $k$ as well as the structural scales $n_y, n_{x|y}, n_{y|x}$.

This BYY DDR is closely related to the BYY DR discussed in Sec.1. The key point of DDR is to reduce the dimension of data ( i.e., $k < d$), but the components of $y$ can be either independent or not, as observed from the difference of $p_{M_y}(y)$ in eq.(1) and eq.(5), while the key point of DR in Sec.1 is to let the components of $y$ become independent and it can be either $k < d$ or $k \geq d$.

To get some deep insights on DDR, we see the linear dimension reduction with gaussian densities—the special case of eq.(5) with $p(y|\xi_j) = G(y, m_y^{(j)}, \Sigma_y^{(j)})$, $p(x - f(y, A_j)|\phi_j) = G(x, A_j y, \Sigma_{x|y}^{(j)})$, $p(y|x, g(x, W_j) = G(y, W_j^T x, \Sigma_{y|x}^{(j)})$, where $x$ is represented by a gaussian mixture in $y$ via different linear mapping $W_j$, which can be regarded as a combined job of data dimension reduction and clustering in $R^k$. We can also regard that $x$ is generated from a gaussian mixture in $y$ via different linear mapping $A_j$.

We further consider the simplest case that $n_y = 1, n_{x|y} = 1, n_{y|x} = 1$ with

$$\begin{aligned} &x = Ay + e_x,\ e_x\ is\ gaussian,\ Ee_x = 0,\ E[e_x e_x^T] = \sigma_{x|y}^2 I_d, \quad p_{M_y}(y) = G(y, 0, \Lambda_y), \\ &p_{M_{x|y}}(x|y) = G(x, Ay, \sigma_{x|y}^2 I_d), \quad p_{M_x}(x) = p_h(x)\ is\ \text{given by eq.[1.2]}. \end{aligned} \tag{6}$$

A particular example of this design has been discussed in Sec.9 of [14] and is shown to be related to the conventional PCA learning. In the following, we consider an even more interesting example that $p_{M_{y|x}}(y|x) = p(y|x)$ is free.

Since $p(y|x)$ is free, the minimization of eq.[1.6] will result in

$$\begin{aligned} &p(x, \theta) = \int G(x, Ay, \sigma_{x|y}^2 I_d) G(y, 0, \Lambda_y) dy = G(x, 0, \sigma_{x|y}^2 I_d + A\Lambda_y A^T), \\ &p(y|x) = \frac{G(x, Ay, \sigma_{x|y}^2 I_d) G(y, 0, \Lambda_y)}{p(x, \theta)}, \quad KL(\Lambda_y, k) = \int_x p_h(x) \ln \frac{p_h(x)}{p(x, \theta)} dx. \end{aligned} \tag{7}$$

Moreover, it is not difficult to see that this $p(y|x)$ is actually in the form

$$\begin{aligned} &p(y|x) = G(y, W^T x, \Sigma_{y|x}), \quad W^T = \Lambda_y A^T (\sigma_{x|y}^2 I_d + A\Lambda_y A^T)^{-1}, \\ &\Sigma_{y|x} = \Lambda_y - W^T (\sigma_{x|y}^2 I_d + A\Lambda_y A^T) W = \Lambda_y - \Lambda_y A^T (\sigma_{x|y}^2 I_d + A\Lambda_y A^T)^{-1} A\Lambda_y. \end{aligned} \tag{8}$$

That is, the situations are actually equivalent when $p_{M_{y|x}}(y|x)$ is either free or gaussian with linear regression.

When $h \to 0$, $\min_\theta KL(\Lambda_y, k)$ with $\theta = \{\sigma_{x|y}^2, A, \Lambda_y\}$ is equivalent to

$$\min_\theta J(\theta_k), J(\theta_k) = \ln |\Sigma_x| + Tr[\Sigma_x^{-1} S_x],\ \Sigma_x = \sigma_{x|y}^2 I_d + A\Lambda_y A^T, \tag{9}$$

with $S_x = \frac{1}{N} \sum_{i=1}^N x_i x_i^T$. We first explore the property of its solution by

$$\begin{aligned} &\nabla_A J(\theta_k) = \Sigma_x^{-1} A\Lambda_y - \Sigma_x^{-1} S_x \Sigma_x^{-1} A\Lambda_y,\ from\ \nabla_A J(\theta_k) = 0, \\ &A^T = A^T \Sigma_x^{-1} S_x,\ and\ S_x = \Phi^T \Lambda_x \Phi, \Phi^T \Phi = I,\ \Lambda_x = diag[\lambda_1^x, \cdots, \lambda_d^x],\ we\ have \\ &A^T \Phi \Lambda_x^{-1} (\sigma_{x|y}^2 I_d + \Phi^T A\Lambda_y A^T \Phi) = A^T \Phi, \quad A^T \Phi = [D|0],\ D = diag[d_1, \cdots, d_k], \\ &[D|0] \Lambda_x^{-1} (\sigma_{x|y}^2 I_d + [D|0]^T \Lambda_y [D|0]) = [D|0], \\ &J(\theta_k) = 0.5[\sum_{j=1}^k \ln (d_j^2 \lambda_j^y + \sigma_{x|y}^2) + (d - k) \ln \sigma_{x|y}^2 + k + \sum_{j=k+1}^d \lambda_j^x / \sigma_{x|y}^2]. \end{aligned} \tag{10}$$

In other words, the solution is not unique and a typical example is that $A$ consists of $k$ eigenvectors of $S_x$, corresponding to the $k$ eigenvalues that minimizes

$$J_1(k) = 0.5[\sum_{j=1}^{k} \ln \lambda_j^x + (d-k)\ln \sigma_{x|y}^2],\ \sigma_{x|y}^2 = \frac{1}{d-k}\sum_{j=k+1}^{d} \lambda_j^x,\ \lambda_j^y = \lambda_j^x - \sigma_{x|y}^2. \quad (11)$$

Unfortunately, directly picking $k$ among $d$ eigenvectors of $S_x$ for eq.(11) is a difficult combinatorial problem. Instead, we propose an iterative algorithm to search a solution on $A^TA = I$. With this constraint, we have $\Sigma_x^{-1} = \sigma_{x|y}^{-2}[I_d - A(\sigma_{x|y}^2\Lambda_y^{-1} + I_k)^{-1}A^T]$ and

$$\begin{aligned}
&J(\theta_k) = \ln|\Lambda_y + \sigma_{x|y}^2 I_k| + \ln|\sigma_{x|y}^2 I_{d-k}| + \sigma_{x|y}^{-2}\{Tr[S_x - (\sigma_{x|y}^2\Lambda_y^{-1} + I_k)^{-1}A^T S_x A]\},\\
&\nabla_{\Lambda_y} J(\theta_k) = (\Lambda_y + \sigma_{x|y}^2 I_k)^{-1} - (\sigma_{x|y}^2 I_k + \Lambda_y)^{-1}A^T S_x A(\sigma_{x|y}^2 I_k + \Lambda_y)^{-1},\\
&\nabla_{\Lambda_y} J(\theta_k) = 0,\ \textit{resulting in}\ A^T S_x A = \Lambda_y + \sigma_{x|y}^2 I_k,\ \textit{or}\ \Lambda_y = A^T S_x A - \sigma_{x|y}^2 I_k,\\
&\nabla_A J(\theta_k) = -2\sigma_{x|y}^{-2} S_x A(\sigma_{x|y}^2\Lambda_y^{-1} + I_k) = -2\sigma_{x|y}^{-2} S_x A\Lambda_y^{-1}(A^T S_x A). \qquad (12)
\end{aligned}$$

Thus, by noticing $\Lambda_y^{-1}A^T S_x A = A^T S_x A\Lambda_y^{-1}$, the gradient of $J(\theta_k)$ with respect to $A$ on the manifold $A^TA = I$ is obtained as

$$\begin{aligned}
&\nabla_A^c = (I - AA^T)\nabla_A J(\theta_k) = -2\sigma_{x|y}^{-2}(S_x A\Lambda_y^{-1} - A\Lambda_y^{-1}A^T S_x A)A^T S_x A,\\
&vec[\nabla_A^c] = -C\,vec[S_x A\Lambda_y^{-1} - A\Lambda_y^{-1}A^T S_x A],\quad C = \sigma_{x|y}^{-2}((A^T S_x A)\otimes I_d). \qquad (13)
\end{aligned}$$

Since $C$ is positive definite due to the positive definite $B$, $S_x A\Lambda_y^{-1} - A\Lambda_y^{-1}A^T S_x A$ is a direction that reduces $J(\theta_k)$ on the manifold $A^TA = I$, and we have the following iterative algorithm for solving $\min_\theta J(\theta_k)$ on this manifold:

$$\begin{aligned}
&A^{new} = A^{old} + \eta(S_x A^{old}\Lambda_y^{-1\ old} - A^{old}\Lambda_y^{-1\ old}A^{old\ T}S_x A^{old}),\\
&\Lambda_y^{new} = A^{old\ T}S_x A^{old} - \sigma_{x|y}^{2\ old} I_k, \qquad (14)\\
&\Sigma_x = \sigma_{x|y}^{2\ old} I_d + A^{old}\Lambda_y^{old}A^{old\ T},\quad \sigma_{x|y}^{2\ new} = \sigma_{x|y}^{2\ old} + \eta Tr[\Sigma_x^{-1}S_x\Sigma_x^{-1} - \Sigma_x^{-1}],
\end{aligned}$$

where the updating for $\sigma_{x|y}^{2\ new}$ is because from the original $J(\theta_k)$ given by eq.(9) we have $\frac{dJ(\theta_k)}{d\sigma_{x|y}^2} = -Tr[\Sigma_x^{-1}S_x\Sigma_x^{-1} - \Sigma_x^{-1}]$. The updating formulae for $\Lambda_y, \Sigma_x$ come directly from eq.(12) and eq.(9). As long as the learning step size $\eta$ is controlled small enough, the algorithm keeps to decrease $J(\theta_k)$ until finally converged.

We can also modify the algorithm eq.(14) into an adaptive one for each $x_i$:

$$\begin{aligned}
&z = A^{old,T}x_i,\quad A^{new} = A^{old} + \eta(x_i z^T\Lambda_y^{-1\ old} - A^{old}\Lambda_y^{-1\ old}zz^T),\\
&\Lambda_y^{new} = (1-\eta)\Lambda_y^{old} + \eta[zz^T - \sigma_{x|y}^{2\ old} I_k], \qquad (15)\\
&\Sigma_x = \sigma_{x|y}^{2\ old} I_d + A^{old}\Lambda_y^{old}A^{old\ T},\quad \sigma_{x|y}^{2\ new} = (1-\eta)\sigma_{x|y}^{2\ old} + \eta(\|\Sigma_x^{-1}x_i\|^2 - Tr[\Sigma_x^{-1}]).
\end{aligned}$$

For a solution $A^*$, $diag[\lambda_1^x, \cdots, \lambda_k^x] = \Lambda_y^* + \sigma_{x|y}^{2\,*} I_k$ by the above algorithms, we can get the best dimension by $k^* = \min_k J_1(k)$ from eq.(11). Also, from eq.(8), $-\int p(y|x)p_h(x)\ln p(y|x)dxdy = 0.5[k + \ln|\Lambda_y(I - A^T(\sigma_{x|y}^2 I_d + A\Lambda_y A^T)^{-1}A\Lambda_y)|]$. By eq.[1.13], after ignoring some terms that are irrelevant to $k$, we can get

$$J_2(k) = 0.5[d\ln(\frac{1}{d-k}\sum_{j=k+1}^{d}\lambda_j^x) + k + \sum_{j=1}^{k}\ln(\lambda_j^x - \frac{1}{d-k}\sum_{j=k+1}^{d}\lambda_j^x)]. \qquad (16)$$

Finally, from the last equation in eq.(10), we get the PCA if $\Lambda_y$ is fixed at any posotive diagonal matrix with different elements, and the *Minor component analysis (MCA)* if $\sigma^2_{x|y}$ is fixed at any large constant. Thus, eq.(14) or eq.(15) act as a unified algorithm for linear dimension reduction that includes the PCA and MCA as special cases.

## 4 BKYY Learning for Three Layer Forward Nets

*Three-layer perceptron* or three-layer forward net is usually trained by the maximum likelihood (ML) or particularly its special case called the *least squares learning* via *back propagation (BP)* technique, which is simple to be understood and thus very popular in the literature. However, its generalization ability depends on the appropriate selection of the number of hidden units or a good regularization technique on its parameter estimation, bases on some heuristics or the complicated upper-bound approximation of generalization error.

Given in Tab.4 are the detailed form of the Fully Matched Yang-based BKYY learning given by eqs.[1.10][1.27]&[1.28] on the cascade architecture —a general form of *three layer forward net* that includes *three layer perceptron* as a special case. The implementation of its parameter learning algorithm involves the integral or summation operations over all the values of $y$ which can be very expensive for a large $k$ except the analytically integrable case that both $p(z|y,\theta_{z|y}), p(y|x,\theta_{y|x})$ are Gaussians and $f(y,W_{y|x}), g(z,W_{z|y})$ are linear. Thus, we need either to develop some efficient algorithm or to make some simplification.

Given in Tab.5 are four such examples. The *first* is exactly the conventional ML learning or particularly the least square learning for a three layer perceptron which can be trained by back-propagation technique. The *second* is obtained by letting $p(z|y,\theta_{z|y})$ be conditioned on the smoothed regression $E(y|x,\theta_{y|x})$ instead of on $y$ directly. These first two examples are equivalent in parameter learning. However, the second also provides an important new result — a new criterion for selecting the best number $k^*$ of the hidden units by $J_2(k)$ that contains an extra term to penalize a large $k$ during its ML fitting on $p_{M_2}(z|x)$. Interestingly, this criterion $J_2(k)$ is much simpler than many of the existing hidden unit selection criteria based on the estimations of the upper bound of generalization error. The *third* example is to use *Mean-Field* approximation on $y$ for all the integral or summation operations over $y$. For real $y$, it turned out to be exactly the same as the first example. However, for binary $y$, the penalizing term appeared in the second example in $J_2(k)$ now appears in $J(\theta_{z|y},\theta_{y|x},k)$ too. In other words, this term not only penalizes $k$ increasing in selecting the best $k$, but also regularizes the parameter learning at a given fixed $k$.

The first three examples are made by either simplification or approximation. In the general cases, by using the general random sampling implementation technique eq.[1.22] and eq.[1.23] given in Sec.3 of [1], we can get a stochastic approximation adaptive algorithm given in Part 4 of Tab.5, for avoiding the integral or summation operations over $y$ during learning.

On the testing phase, three ways of implementing mapping $x \to z$ are given

in Part 5 of Tab.5. Moreover, via randomly sampling, we can also get new criteria for selecting the best number $k^*$ of the hidden units.

## 5 BKYY Learning for Mixture of Experts

As discussed previously, the cascade architecture of three layer perceptron encounters the integral or summation over $y$, which causes an impractical cost unless $y$ takes only a few values. However, if $y$ takes just a few values, the representation ability of the network is limited since $y$ functions as a bottle-neck that the entire information flow must go through.

In the case of the Fully Matched BKYY learning, when $p_{M_{z|x,y}}(z|x,y) \neq p_{M_{z|y}}(z|y)$, eq.[1.27] will become $p_{M_2}(x,z) =$

$$\int p_{M_{z|y}}(z|x,y)p_{M_2^L}(x,y)dy = \begin{cases} \int p_{M_{z|y}}(z|x,y)p_{M_{x|y}}(x|y)p_{M_y}(y)dy, & \textit{Ying based,} \\ \int p_{M_{z|y}}(z|x,y)p_{M_{y|x}}(y|x)p_{M_x}(x)dy, & \textit{Yang based;} \end{cases}$$

$$p_{M_2}(z|x) = \frac{p_{M_2}(x,z)}{\int_z p_{M_2}(x,z)dz} = \begin{cases} \int_y p_{M_{z|y}}(z|x,y)p_{M_{x|y},M_y}(y|x)dy, & \textit{Ying based,} \\ \int_y p_{M_{z|y}}(z|x,y)p_{M_{y|x}}(y|x)dy, & \textit{Yang based;} \end{cases}$$

$$p_{M_{x|y},M_y}(y|x) = \frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{\int_y p_{M_{x|y}}(x|y)p_{M_y}(y)dy}. \tag{17}$$

In this case, each $p_{M_{z|x,y}}(z|x,y)$ itself builds a direct link $x \to z$, gated via the internal variable $y$ such that a weighted mixture $p_{M_2}(z|x)$ is formed in a *parallel architecture*, with a gate by $p_{M_{y|x}}(y|x)$ for a Yang based system and $p_{M_{x|y},M_y}(y|x)$ for a Ying based system. Although we still encounter the summation over all the values of $y$, here $y$ only takes a small number of values. Because each $p_{M_{z|x,y}}(z|x,y)$ has a direct link $x \to z$ and $y$ only functions as a gate that weights information flows from different experts, we can trade-off the complexity of $y$ and the structural complexity of each $p_{M_{z|x,y}}(z|x,y)$ such that the number of values that $y$ should take are significantly fewer than it should take on the cascade architecture. This is an advantage the cascade architecture does not have.

In Tab.6, we let $y$ take only $k$ discrete values. From Part 6, we see that the Yang based system is actually the model of *Mixture of Experts* [9, 10, 5, 17, 18] with each $p_{M_{z|x,y}}(z|x,y)$ being an expert net and $p_{M_{y|x}}(y|x)$ being the so called *softmax* gating net, and that the Ying based system is actually the alternative model of *Mixture of Experts* proposed in [17, 18], with $p(y|x) = p(x|\phi_j)p(y_j = 1)/\sum_{j=1}^k p(x|\phi_j)p(y_j = 1)$ as an alternative model for the gating net. Moreover, the algorithm in Part 4 for the Yang and Ying based systems is actually the EM algorithm for the original model for mixture of experts [7, 9] and the EM algorithm for the alternative model of mixture experts proposed in [17, 18], respectively. This fact can also directly be observed from $J(\theta,\phi,k) = -L_{z|x}$ in Part 3. Actually Part 3 indicates that $\min_{\{\theta,\phi,k\}} J(\theta,\phi,k)$ is equivalent to the maximum likelihood learning on $p_{M_2}(z|x)$ or $p_{M_2}(z,x)$.

One inconvenience of using EM on the original model is the nonlinearity of *softmax* gating net, which makes the maximization with respect to the parameters in the gating net become nonlinear and unsolvable analytically even for

**Table 6. BKYY Learning for Mixture of Experts**

**1. Variable Types**

$y = [y_1, \cdots, y_k]$ with $y_r \in \{0, 1\}, \sum_{r=1}^{k} y_r = 1$, $z$ is the same as in Tab.4

**2. Architecture Design**

$p_{M_{y|x,z}}(y|x,z) = \sum_{j=1}^{k} y_j p(y_j = 1|x,z)$ and $p_{M_y}(y) = \sum_{j=1}^{k} y_j p(y_j = 1)$ are free

| | |
|---|---|
| $p_{M_{z|x,y}}(z|x,y) = \sum_{j=1}^{k} y_j p(z|x,\theta_j)$ | $p_{M_{y|x}}(y|x) = \sum_{j=1}^{k} y_j p(y_j = 1|x,\phi)$ |
| $p(z|x,\theta_j) =:$ | $p(y_j = 1|x,\phi) = \mu_j / \sum_{j=1}^{k} \mu_j,\ \mu_j = f_j(x,V)$ |
| $\begin{cases} \prod_{r=1}^{m} \pi_{j,r}^{z_r}(1-\pi_{j,r})^{1-z_r}, & \text{Binary-F } z \\ \sum_{r=1}^{m} z_r \pi_{j,r} / \sum_{r=1}^{m} \pi_{j,r}, & \text{Binary-E } z, \\ G(z, g(x,W_j), \Sigma_j), & \text{Real } z. \end{cases}$ | $p(x|\phi_j) = \begin{cases} G(x, m_j, \Pi_j), & \text{Real } x, \\ \prod_{r=1}^{d} q_{j,r}^{x_r}(1-q_{j,r}^{x_r})^{1-x_r}, & \text{binary } x. \end{cases}$ |
| $\pi_{j,r}(x,W_j) = g_r(x,W_j),$ | $p_{M_{x|y}}(x|y) = \sum_{j=1}^{k} y_j p(x|\phi_j),$ |
| $g(x,W_j) = [g_1(x,W_j), \cdots, g_m(x,W_j)]$ | $f(x,V) = [f_1(x,V), \cdots, f_k(x,V)]$ |
| e.g., $g_j(x,W_j) = s(e_j^T(W_j x))$ | e.g., $f_j(x,V) = s(e_j^T(Vx))$ |

**3. Learning, i.e.,** $\min_{\{\theta,\phi,k\}} J(\theta,\phi,k),\ \theta = \{\theta_j, j = 1, \cdots, k\},\ \phi = \{\phi_j, j = 1, \cdots, k\}$

| The Yang based system | The Ying based system |
|---|---|
| $J(\theta,\phi,k) = \begin{cases} -H_{y|x,z} - L_{z|x,y} - L_{y|x}, \\ -L_{z|x}; \end{cases}$ | $J(\theta,\phi,k) = \begin{cases} -H_{y|x,z} - L_{z|x,y} - L_{x|y} + H_y, \\ -L_{x,z}; \end{cases}$ |
| $p(y_j = 1|x,z) = \frac{p(z|x,\theta_j)p(y_j=1|x,\phi)}{p_{M_2}(z|x)}$ | $p(y_j = 1|x,z) = \frac{p(z|x,\theta_j)p(x|\phi_j)p(y_j=1)}{p_{M_2}(x,z)}$ |
| $p_{M_2}(z|x) = \sum_{j=1}^{k} p(z|x,\theta_j)p(y_j = 1|x,\phi)$ | $p_{M_2}(x,z) = \sum_{j=1}^{k} p(z|x,\theta_j)p(x|\phi_j)p(y_j = 1)$ |
| $L_{y|x} = \int L_{y|x}(x,z) p_{M_1^L}(z,x)dxdz,\ L_{y|x}(x,z)$ | $L_{x|y} = \int L_{x|y}(x,z) p_{M_1^L}(z,x)dxdz$ |
| $= \sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(y_j = 1|x,\phi)$ | $L_{x|y}(x,z) = \sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(x|\phi_j)$ |
| $L_{z|x} = \int p_{M_1^L}(z,x) \ln p_{M_2}(z|x)dxdz$ | $L_{x,z} = \int p_{M_1^L}(z,x) \ln p_{M_2}(x,z)dxdz$ |

$H_{y|x,z} = \int H_{y|x,z}(x,z) p_{M_1^L}(z,x)dxdz,\ H_{y|x,z}(x,z) = -\sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(y_j = 1|x,z)$

$L_{z|x,y} = \int L_{z|x,y}(x,z) p_{M_1^L}(z,x)dxdz,\quad L_{z|x,y}(x,z) = \sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(z|x,\theta_j)$

$H_y = -\sum_{j=1}^{k} p(y_j = 1) \ln p(y_j = 1),\quad p(y_j = 1) = \int p(y_j = 1|x,z) p_{M_1^L}(z,x)dxdz$

When $p_{M_1^L}(z,x) = p_{M_x}(x) p_{M_{z|x}}(z|x)$ given by eq.[1.2] and eq.[1.8],

the same as in Tab.5 except $L_{z|y}$ is replaced by $L_{z|x,y} = \sum_{(x,z)\in D_{x,z}} L_{z|x,y}(x,z)$.

**4. Parameter Learning Algorithm at a fixed $k$**

Step 1: Fix $\theta, \phi$, get $p(y_j = 1|x,z)$ above

| | |
|---|---|
| Step 2: | Step 2: $p(y_j = 1) =$ |
| For $j = 1, \cdots, k$ | $\frac{1}{\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} p(y_j = 1|x,z)$ |
| update $\theta_j^{new} = arg\max_{\theta_j} L_{z|x,y}$ | $\theta_j^{new} = arg\max_{\theta_j} L_{z|x,y}, j = 1, \cdots, k,\ m_j^{new} =$ |
| $\phi^{new} = arg\max_{\phi} L_{y|x}$ | $\frac{1}{p(y_j=1)\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} p(y_j = 1|x,z)x$ |
| e.g., moving one step along the | $\Sigma_j^{new} = \frac{1}{p(y_j=1)\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} \times$ |
| gradient ascent direction. | $p(y_j = 1|x,z)[x - m_j^{new}][x - m_j^{new}]^T$ |

**5. Model Selection** (*selecting the number $k$ of hidden unit*)

With $\theta^*$, $\phi^*$ obtained by the above algorithm, select $k^*$ by

From eqs.[1.13&14], by either $J_1(k) = J(\theta^*, \phi^*, k)$ or $J_2(k) = J_1(k) + H_{y|x,z}|_{\{\theta^*,\phi^*\}}$

When $p(z|x,\theta_j)$ is gaussian, they actually take the specific form:

$J_1(k) = \sum_{(x,z)\in D_{x,z}} \sum_{j=1}^{k} p^*(y_j = 1|x,z) \ln p^*(y_j = 1|x,z) + J_2(k),$

| | |
|---|---|
| $J_2(k) = -\sum_{(x,z)\in D_{x,z}} \sum_{j=1}^{k} \times$ | $J_2(k) = \ln p(y_j = 1|x,\phi^*) \sum_{j=1}^{k} \times$ |
| $[-p^*(y_j = 1|x,z) \ln p(y_j = 1|x,\phi^*) +$ | $[-p^*(y_j = 1) \ln p^*(y_j = 1) + 0.5p^*(y_j = 1)$ |
| $0.5p^*(y_j = 1) \ln \lvert\Sigma_j^*\rvert]$ | $\times \ln \lvert\Pi_j^*\rvert + 0.5p^*(y_j = 1) \ln \lvert\Sigma_j^*\rvert]$ |

**6. The Output of Nets** $p_{M_2}(z|x) = \sum_{j=1}^{k} p(z|x,\theta_j)p(y_j = 1|x,\phi)$

For Ying based case, $p(y_j = 1|x,\phi) = p(x|\phi_j)p(y_j = 1) / \sum_{j=1}^{k} p(x|\phi_j)p(y_j = 1)$

**Table 7. BKYY Learning for Normalized RBF Nets**

**1. Variable Types and Architecture Design**

All are the same as the Ying-based system in Tab.6, with specifically:

$$p(z|x,\theta_j) = G(z, c_j, \Sigma_j), \text{ or } p(z|x,\theta_j) = G(z, W_j^t x + c_j, \Sigma_j), \quad p(y_j = 1) = \frac{\sqrt{|\Pi_j|}}{\sum_{j=1}^k \sqrt{|\Pi_j|}}$$

**2. The Batch Way EM Algorithm at a fixed $k$**

Step 1: $r_j^{old} = \begin{cases} c_j^{old}, & \text{for a NRBF net,} \\ (W_j^{old})^t x + c_j^{old}, & \text{for an ENRBF net;} \end{cases}$

$$h(j|x) = p(y_j = 1|x,z) = \frac{e^{-0.5[x-m_j^{old}]^T[\Pi_j^{old}]^{-1}[x-m_j^{old}]}_{G(z, r_j^{old}, \Sigma_j^{old})}}{\sum_{j=1}^k e^{-0.5[x-m_j^{old}]^T[\Pi_j^{old}]^{-1}[x-m_j^{old}]}_{G(z, r_j^{old}, \Sigma_j^{old})}}$$

Step 2: First, for both NRBF and ENRBF nets, get $N_e = \sum_{(x,z)\in D_{x,z}} h(j|x)$, update

$$m_j^{new} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)x, \quad \Pi_j^{new} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)[x - m_j^{new}][x - m_j^{new}]^T;$$

$$\text{NRBF net: } c_j^{new} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)z,$$

$$\Sigma_j^{new} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)(z - c_j^{new})(z - c_j^{new})^T;$$

$$\text{ENRBF net: } \quad Ez_j = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)z,$$

$$R_{xz} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)[z - Ez][x - (m_j)^{new}]^T,$$

$$W_j^{new} = [(\Pi_j)^{new}]^{-1} R_{xz}, c_j^{new} = Ez_j - (W_j^{new})^t (m_j)^{new}$$

$$\Sigma_j^{new} = \frac{1}{N_e}\sum_{(x,z)\in D_{x,z}} h(j|x)[z - (W_j^{new})^t x - c_j^{new}][z - (W_j^{new})^t x - c_j^{new}]^T.$$

$$\text{When } \Sigma_j = \sigma_j^2 I, \ \sigma_j^2 = \frac{1}{N_e}\sum_{j=1}^k \sum_{(x,z)\in D_{x,z}} h(j|x)\|z - r_j^{new}\|^2,$$

**3. The Adaptive EM Algorithm at a fixed $k$**

Step 1: get the same $h(j|x)$ as in the batch way case,

$$\text{update } \alpha_j^{new} = (1-\eta_0)\alpha_j^{old} + \eta_0 h(j|x),$$

$$\text{then get } I(j|x_i) = \begin{cases} 1, & \text{if } j = arg\min_r\{-\log h(r|x_i)\}, \\ 0, & \text{otherwsie.} \end{cases}, \ \eta_{j,i} = \eta_0 h(j|x_i)/\alpha_j;$$

Step 2 : Update $m_j^{new} = m_j^{old} + \eta_{j,i}(x_i - m_j^{old}),$ ,

$$\Pi_j^{new} = (1-\eta_{j,i})\Pi_j^{old} + \eta_{j,i}(x_i - m_j^{old})(x_i - m_j^{old})^T$$

$$\text{NRBF net, } c_j^{new} = c_j^{old} + \eta_{j,i}(z_i - c_j^{old}), \ \Sigma_j^{new} = (1-\eta_{j,i})\Gamma_j^{old} + \eta_{j,i}(z_i - c_j^{new})(z_i - c_j^{new})^T;$$

$$\text{ENRBF net, } Ez_j^{new} = Ez_j^{old} + \eta_{j,i}(z_i - Ez_j^{old}), \ c_j^{new} = Ez_j^{new} - W_j^{old\ T} m_j^{old};$$

$$\Sigma_j^{new} = (1-\eta_{j,i})\Sigma_j^{old} + \eta_{j,i}(z_i - W_j^{old\ T} x_i - c_j^{new})(z_i - W_j^{old\ T} x_i - c_j^{new})^T,$$

$$W_j^{new} = W_j^{old} + \eta_{j,i}(z_i - W_j^{new\ T} x_i - c_j^{new})x_i^T$$

**4. Model Selection** (*selecting the number $k$ of radial basis functions*)

$$J_1(k) = \sum_{(x,z)\in D_{x,z}} \sum_{j=1}^k p^*(y_j = 1|x,z) \ln p^*(y_j = 1|x,z) + J_2(k),$$

$$J_2(k) = \ln \sum_{j=1}^k \sqrt{|\Pi_j^*|} + \sum_{j=1}^k \frac{\sqrt{|\Pi_j^*|}}{\sum_{j=1}^k \sqrt{|\Pi_j^*|}} \ln \sqrt{|\Sigma_j^*|}$$

piecewise linear case. An algorithm called *Iteratively Reweighted Least Squares* (IRLS) is proposed for the nonlinear optimization [9, 10]. However, IRLS is a Newton-type algorithm and thus needs some safeguard measures for convergence, which is a big difference from the guaranteed convergence of the pure EM algorithm. This inconvenience has been overcome by the alternative model. As shown in Part 3, the learning on the gate can be made analytically.

Another important open problem of using both the original and alternative mixture of experts is how to select the number of experts. Based on the BKYY

learning theory, we proposed a criteria for this task by Part 5 in Tab.6. We can intuitively observe $J_2(k)$. As $k$ increases, I ts first term will increase which trades off the monotonically decreasing by the second term for a best $k^*$.

## 6 BKYY Learning for Normalized RBF Nets

We consider an interesting special case of the Ying-based system in Tab.6, with a specific architecture design by Part 1 in Tab.7. In this case, from $p_{M_2}(z|x)$ by Part 6 in Tab.6, we have $E(z|x) = \int p_{M_2}(z|x)dz$ given by

$$E(z|x) = \begin{cases} \sum_{j=1}^{k} c_j \phi(x - m_j), \\ \sum_{j=1}^{k} (W_j^t x + c_j)\phi(x - m_j) \end{cases}, \quad \phi(u) = \frac{e^{-0.5u^T \Pi_j^{-1} u}}{\sum_{j=1}^{k} e^{-0.5u^T \Pi_j^{-1} u}}. \tag{18}$$

We can easily find that they are just the standard normalized RBF (NRBF) net and extended normalized RBF (ENRBF) nets with $k$ hidden units, which have been widely studied in the literature, see the reference list of [19]. In other words, *the maximum likelihood (ML) learning on the normalized RBF nets is a special case of the ML learning on the alternative model of mixture of experts by the EM algorithm.*

In the existing literature, RBF net is expected to be trained by the least square learning — a special case of ML learning. However, due to the difficulty of determining the parameters $\Pi_j$, $m_j$ of basis functions, in practice the learning is usually made approximately by two separate steps. First, $\Pi_j = (\sigma_j^g)^2 I$ with $(\sigma_j^g)^2$ being estimated roughly and heuristically, and some cluster analysis (e.g., the k-means algorithm) is used to group data set $D_x = \{x_i\}_{i=1}^N$ into $k$ clusters, and then use the cluster centers as $m_j, j = 1, \cdots, k$. Second, the output layer parameters $c_j, j = 1, \cdots, k$ are determined by the least square learning. By this two-stage training approach, the centers $m_j, j = 1, \cdots, k$ are obtained directly from input data without considering how to get the best regression $E(z|x)$.

Moreover, there is also another major problem — how to select the number of basis functions, which will affect the performance considerably. In [19], via setting up the connections between RBF nets and kernel regression estimators, a number of theoretical results have been obtained for the upper bounds of convergence rate of the approximation error with respect to the number of basis functions, and the upper bounds for the pointwise convergence rate and $L_2$ convergence rate of the best consistent estimator with respect to both the samples and the number of basis functions. However, these theoretical results are not directly usable in practice. Rival Penalized Competitive Learning (RPCL) is able to automatically select the number of clusters and thus suggested for RBF nets [20]. However, although it experimentally works well, RPCL is a heuristic approach and still in lack of theoretical justification.

From the connection eq.(18) between the BKYY Ying-based learning for mixture of experts, we can solve all the above problems. First, we can directly use the EM algorithm in Tab.6 to train the NRBF and ENRBF nets to get all the parameters $\Pi_j$, $m_j$, $W_j, c_j, \Sigma_j$ such that a globally more optimal solution is

obtained. Specifically, the detailed form of the EM algorithm is given in Tab.7, together with its adaptive variants. Second, we can select the number $k$ of radial basis functions by the criterion by Part 4 in Tab.7.

## 7 BCYY Learning and Partially Matched Learning

**(1) BYY Learning with Convex Divergence**

Instead of eq.[1.7], we use the *Convex Divergence* eq.[1.6] for the *Bayesian Convex Ying-Yang (BCYY)* learning. In this case, most of the results given in Theorems 3.1-3.4 in [1] do not hold anymore. However, we can still use eq.[1.17] and the general implementation technique by Sec.3 in [1] for the minimization of $F_{two}(M_1, M_2)$. Here, we only discuss such a learning for the mixture of experts given in Tab.6.

The key point is to still use the same Step 1 in Tab.6, and then in Step 2, we directly maximize

$$L_{z,x} = \int_{x,z} p_{M_1^L}(z,x) f(p_{M_2}(x,z)) dxdz, \quad for\ a\ Ying\ based\ system,$$
$$L_{z|x} = \int_{x,z} p_{M_1^L}(z,x) f(p_{M_2}(z|x)) dxdz, \quad for\ a\ Yang\ based\ system. \tag{19}$$

with $p_{M_2}(x,z), p_{M_2}(z|x)$ still given by Part 3 in Tab.6, resulting in:

Step 2 (Yang -based ): get $w(j,x) = f'(p_{M_2}(z|x)) p_{M_2}(z|x) p(y_j = 1|x,z)$, $f'(u) = \frac{df(u)}{du}$, update $\theta_j^{new}$ via solving
$\sum_{(x,z)\in D_{x,z}} w(j,x_i) \frac{d \ln p(z|x,\theta_j)}{d\theta_j} = 0, \sum_{(x,z)\in D_{x,z}} w(j,x_i) \frac{d \ln p(y_j=1|x,\phi)}{d\phi} = 0.$

Step 2 (Ying -based ): get $w(j,x) = f'(p_{M_2}(z,x)) p_{M_2}(z,x) p(y_j = 1|x,z)$, update $\theta_j^{new}$ via solving $\sum_{(x,z)\in D_{x,z}} w(j,x) \frac{d \ln p(z|x,\theta_j)}{d\theta_j} = 0$, and then get
$$m_j^{new} = \frac{1}{p(y_j=1)\# D_{x,z}} \sum_{(x,z)\in D_{x,z}} w(j,x) x,$$
$$\Sigma_j^{new} = \frac{1}{p(y_j=1)\# D_{x,z}} \sum_{(x,z)\in D_{x,z}} w(j,x)[x - m_j^{new}][x - m_j^{new}]^T.$$

For the same reason as in [15], when $f(u)$ is monotonically increasing for positive $u$, e.g., $f(u) = u^\beta$, $0 < \beta < 1$, the learning will give a more robust estimation for data of multi-modes with outliers or high overlap between clusters.

**(2) Partially Matched BKYY Learnings**

We relax the satisfaction of eq.[1.24] and consider the learning eq.[1.9] on the architecture of mixture of experts:

(1) **Yang based system**. In this case, $p_{M_{x|y}}(x|y)$ and $p_{M_y}(y)$ are both free. From Theorem 4 in [1], $L_{y|x}(x,z)$ in $L_{y|x}$ of Tab.6 should be replaced by

$$L_{y|x}(x,z) = \sum_{j=1}^{k} p(y_j = 1|x,z)[\ln p(y_j = 1|x,\phi) + \ln(1 + \frac{p(y_j = 1|x,z)}{p(y_j = 1|x,\phi)})], \tag{20}$$

for parameter learning. Moreover, for model selection we need to add to $J_1(k)$ an extra term given by

$$J^a(k) = \sum_{(x,z)\in D_{x,z}} \sum_{j=1}^{k} p^*(y_j = 1|x,z) \ln (1 + \frac{p^*(y_j = 1|x,z)}{p(y_j = 1|x,\phi)}). \quad (21)$$

(2) **Ying based system**. In this case, $p_{M_{y|x}}(y|x) = p(y|x)$ and $p_{M_y}(y)$ are free. From Theorems 2 & 3 in [1], we have

$$p(y_j = 1|x) = p(x|\phi_j)p(y_j = 1)/\sum_{j=1}^{k} p(x|\phi_j)p(y_j = 1),$$
$$p(y_j = 1) = 0.5[\frac{1}{\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} p(y_j = 1|x,z) + \frac{1}{\#D_x} \sum_{x\in D_x} p(y_j = 1|x)]. \quad (22)$$

Therefore in Tab.6 we should replace $p(y_j = 1|x,z)$ in $L_{x|y}(x,z)$ by $p(y_j = 1|x,z) + p(y|x)$, let $p(y_j = 1)$ in $H_y$ given by eq.(22), and replace $H_{y|x,z}(x,z)$ by $H_{y|x,z}(x,z) = -\sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(y_j = 1|x,z) - \sum_{j=1}^{k} p(y_j = 1|x,z) \ln p(y_j = 1|x)$
Then we modify the algorithm in Tab.6 into:
Step 1: Run the original step 1 plus getting $p(y_j = 1|x)$ by eq.(22),
Step 3: Get $p(y_j = 1)$ by eq.(22), update $\theta_j^{new}$ as in the original step 2 and update

$$m_j^{new} = \frac{1}{2p(y_j=1)\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} [p(y_j = 1|x,z) + p(y_j = 1|x)]x$$
$$\Sigma_j^{new} = \frac{1}{2p(y_j=1)\#D_{x,z}} \sum_{(x,z)\in D_{x,z}} [p(y_j = 1|x,z) + p(y_j = 1|x)][x - m_j^{new}][x - m_j^{new}]^T.$$

Also, the selection of $k$ should be made by (with $h_j^* = p^*(y_j = 1|x,z)$)

$$J_1(k) = \sum_{(x,z)\in D_{x,z}} \sum_{j=1}^{k} [h_j^* \ln h_j^* + p^*(y_j = 1|x) \ln p^*(y_j = 1|x)] + J_2(k), \quad (23)$$

$$J_2(k) = \sum_{j=1}^{k} [-2p^*(y_j = 1) \ln p^*(y_j = 1) + p^*(y_j = 1) \ln |\Pi_j| + 0.5 \sum_{(x,z)\in D_{x,z}} h_j^* \ln |\Sigma_j^*|].$$

## 8 Conclusions

This paper has further interpreted the theory through its uses on developing models and algorithms for dependence reduction, ICA and blind source separation, linear dimension reduction, supervised classification and regression by feed forword net, mixture of experts and RBF nets.

## References


1. Xu, L., " Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach (II): From Unsupervised Learning to Supervised Learning and Temporal Modeling ", in the same proceedings, pp25-42. .

2. S.-I. Amari, A. Cichocki, H. Yang, "A new learning algorithm for blind separation of sources", in *Advances in NIPS 8*, MIT Press, 1996, 757-763.
3. H.B. Barlow, "Unsupervised learning", Neural Computation, **1**, 295-311, 1989.
4. A.J. Bell and T.J. Sejnowski, "An information-maximatization approach to blind separation and blind deconvolution", *Neural Computation* **7** (1995) 1129-1159.
5. Dempster, A., Laird, N. M., & Rubin, D. B., "Maximum-likelihood from incomplete data via the EM algorithm", *J. Royal Statistical Society, B*, **39** (1), 1-38, (1977).
6. M.Gaeta and J.-L Lacounme, " Source Separation without a priori knowledde: the maximum likelihood solution", in *Proc. EUSIPCO90*, pp621-624, 1990.
7. Jacobs, R.A., Jordan, M. I., Nowlan, S.J., & Hinton, G. E., "Adaptive mixtures of local experts", *Neural Computation, 3*, (1991), 79-87.
8. C.Jutten, "From source separation to Independent component analysis: An introduction to special session", *Proc. of 1997 European Symp. on Artificial Neural Networks*, Bruges, April 16-18, pp243-248.
9. Jordan, M. I. & Jacobs, R.A., "Hierarchical mixtures of experts and the EM algorithm", *Neural Computation* **6**(1994), 181-214.
10. Jordan, M. I. & Xu, L., "Convergence results for the EM approach to mixtures of experts", *Neural Networks*, **8**(9)(1995), 1409–1431.
11. Moody, J. & Darken, J., "Fast learning in networks of locally-tuned processing units", *Neural Computation*, **1** (1989), pp281-294.
12. D.T.Pham, P. Garat and C. Jutten, "Separation of a mixture of independent sources through a maximum likelihood approach", in *Singal Processing VI: Theories and Applications*, J. Vandewalle et al (eds), Elsevier Science Pub., 1992, pp771-774.
13. A.Taleb and C.Jutten, "Nonlinearity source separation: the post-nonlinear mixtures", the same proceedings of [8], pp279-284 , 1997.
14. Xu, L., "Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach: (I) Unsupervised and Semi-Unsupervised Learning", Invited paper, ", S. Amari and N. Kassabov eds., *Brain-like Computing and Intelligent Information Systems*, 1997, Springer-Verlag, pp241-274.
15. Xu, L., "Bayesian Ying-Yang Machine, Clustering and Number of Clusters", in press, *pattern Recognition Letters*, 1997.
16. L. Xu, C.C. Cheung, J. Ruan, and S.-I. Amari, "Nonlinearity and Separation Capability: Further Justification for the ICA Algorithm with A Learned Mixture of Parametric Densities", the same proceedings of [8], pp291-296, 1997.
17. Xu, L., Jordan, M. I. & Hinton, G. E., "An alternative model for mixtures of experts", in *Advances in Neural Information Processing Systems*, J. D. Cowan, et al, Eds., MIT Press, Cambridge MA, (1995), pp. 633–640.
18. Xu, L. & Jordan, M. I., "A modified gating network for the mixtures of experts architecture", in *Proc. WCNN94*, San Diego, ppII405–II410.
19. Xu, L., Krzyzak, A. & Yuille, A.L., " On Radial Basis Function Nets and Kernel Regression: Statistical Consistency, Convergence Rates and Receptive Field Size", *Neural Networks*, **5**(4)(1994), 609-628.
20. Xu, L., A.Krzyzak, and E.Oja, Rival Penalized "Competitive Learning for Clustering Analysis, RBF net and Curve Detection", *IEEE Trans. on Neural Networks 4*(4)(1993), 636-649.