Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach (I): for Unsupervised and Semi-Unsupervised 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/

An Invited book chapter, S. Amari and N. Kassabov eds.,
Brain-like Computing and Intelligent Information Systems,
1997, Springer-Verlag , pp241-274.

Abstract. A unified statistical learning approach called *Bayesian Ying-Yang (BYY) system and theory* has been developed by the present author in recent years. This paper is the first part of a recent effort on systematically summarizing this theory. In this paper, we show how the theory functions as a general theory for unsupervised learning and its semi-unsupervised extension on *parameter learning, regularization, structural scale or complexity selection, architecture design and data sampling.* Specifically, it is shown how the general theory provides new theories for *unsupervised pattern recognition and clustering analysis, factorial encoding, data dimension reduction, and independent component analysis*, such that not only several existing popular unsupervised learning approaches, (e.g., finite mixture with the EM algorithm, K-means clustering algorithm, Helmholtz machine, principal component analysis plus various extensions, Informax and minimum mutual information approaches for independent component analysis, ..., etc), are unified as special cases with new insights and several new results, but also a number of new unsupervised learning models are obtained. In a sister paper [12], this theory is further shown to function as a general theory for supervised learning too, from which we get new theories for supervised classification and regression such that the existing approaches for multilayer net, mixtures-of-experts, and radial basis function nets are unified as special cases, with not only new insights and new learning algorithms but also new selection criteria for the number of hidden units and experts. In another sister paper [13], this theory is further shown to function as a general theory for learning on time series also, not only with the hidden Markov model and the linear state space based Kalman filter as special cases, but also with several temporal learning models and algorithms obtained.

1. Introduction

Perception and Association are two primary tasks of a brain-like system for intelligent interactions between the inner activities in the system and the

external environment that the system is facing. Given a pattern or object x in an external environment X, the task of *perception* is to build a representation y for x in the internal representation domain Y of the intelligent system as the foundation of subsequent high level intelligent activities. According to the difference of the numeric type of y, the task is usually called *pattern recognition, encoding, feature extraction ..., etc*, respectively. Given x in X and z in another external environment Z, the task of *association* is to build an associative link from x to z via the intelligent system, which represents the system's response to external environment under a given external input. According to the numeric type of z, the task is usually called *classification, regression, function approximation, control action, ..., etc*, respectively. The intelligent system's ability of implementing the two tasks are obtained via *unsupervised and supervised learning*.

In the past two decades, various models and theories have been proposed for each of the two primary intelligent tasks and related learning problems. In the recent three years, a new statistical approach called *Bayesian Ying-Yang (BYY) system and theory* has been developed by the present author[12-16, 20-23, 25-27]. Not only the two intelligent tasks and their related learning issues can be systematically described in a single framework, but also several existing major statistical models and theories for both unsupervised learning and supervised learning can be naturally unified with deep insights and cross-fertilizations. Furthermore, as stated in the previous *abstract* section, this approach also provides a general theory on parameter learning, regularization, structural scale or complexity selection, architecture design and data sampling in various major areas of statistical learning, with a considerable number of interesting new results obtained.

This paper concentrates on the BYY learning system and theory for implementing various perception tasks via unsupervised learning and its extension called semi-unsupervised learning, with the fundamental issues introduced in Secs 2-6 and new systems and theories proposed in Secs. 7-10 for unsupervised pattern recognition and clustering analysis, factorial encoding, data dimension reduction, and independent component analysis, respectively. A sister paper [12] will concentrate on introducing the BYY learning system and theory for various association tasks via supervised learning. Moreover, how this general theory works for learning on time series will be given in another sister paper [13].

2. Basic Bayesian Ying-Yang System and Its Learning

The perception tasks can be summarized into the mapping $x \in X \to y \in Y$, which is described by the conditional distribution $p(y|x)$ in the probabilistic formulation. $p(y|x)$ is implemented by a device or passage $M_{y|x}$.

As shown in Fig.2.1, in our framework the learning of this $p_{M_{y|x}}(y|x)$ is not independent, but regarding as a part of the problem of estimat-

ing joint distribution $p(x,y), x \in X, y \in Y$. Under the Bayesian framework, we have two complementary representations $p(x,y) = p(y|x)p(x)$ and $p(x,y) = p(x|y)p(y)$. We use two sets of models $M_1 = \{M_{y|x}, M_x\}$ and $M_2 = \{M_{x|y}, M_y\}$ to implement each of the two representations:

$$p_{M_1}(x,y) = p_{M_{y|x}}(y|x)p_{M_x}(x), \quad p_{M_2}(x,y) = p_{M_{x|y}}(x|y)p_{M_y}(y). \tag{2.1}$$

We call M_x as a Yang/(visible) model, which describes $p(x)$ on the visible domain X, and M_y as a Ying[1]/(invisible) model which describes $p(y)$ on the invisible domain Y. Also, we call the passage $M_{y|x}$ for the flow $x \to y$ as a *Yang*/(male) passage since it performs the task of transferring a pattern/(a real body) into a code/(a seed). We call a passage $M_{x|y}$ for the flow $y \to x$ as a *Ying*/(female) passage since it performs the task of generating a pattern/(a real body) from a code/(a seed). Together, we have a *YANG* machine M_1 to implement $p_{M_1}(x,y)$ and a *YING* machine M_2 to implement $p_{M_2}(x,y)$. A pair of YING-YANG machines is called a YING-YANG pair or a Bayesian YING-YANG system. Such a formalization compliments to a famous Chinese ancient philosophy that *every entity in universe involves the interaction between YING and YANG*.

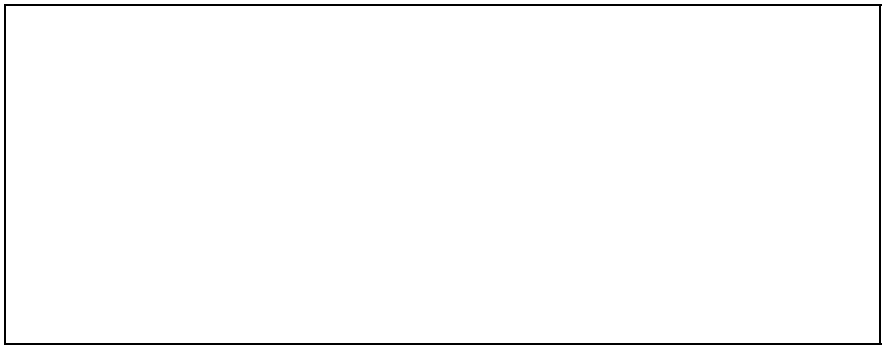

Fig. 2.1. The joint input-representation spaces X, Y and the Ying-Yang system.

The task of specifying a Ying-Yang system is called *learning* in a broad sense, which consists of the following four levels of specifications:

(1) **Representation Domain Y and Its Complexity**

We will encounter both the cases that y is discrete as in Item 2.1 and Item 2.2 and that y is real as in Item 2.3. Strictly speaking, we can only use $p(.)$ as in $p(y), p(y|x), p_{M_{y|x}}(y|x)$, and $p_{M_y}(y)$ to denote densities when y is real. When y is discrete, they should be probabilities instead of densities and we should use $P(.)$ to replace $p(.)$. However, for convenience, even for a discrete y we still use $p(.)$ for denoting probabilities. Readers may identify the difference according to whether y

[1] It should be "Yin" in the current Mainland Chinese spelling system. However, I prefer to use "Ying" for the beauty of symmetry.

is real or discrete. Moreover, in the cases that x can be either real or discrete, we also use the same convention for $p(x), p_{M_x}(x)$, and $p_{M_{x|y}}(x|y)$.

With this preparation, we are ready to introduce the following choices for the design of Y domain, corresponding to different types of $x \to y$ tasks:

- **Item 2.1** *Y consists of a number of discrete labels* or equivalently $y \in [1, 2, \cdots, k_r]$. That is, x is mapped into one of k_r labels, which is usually called *pattern recognition* in general or *classification* in particular. Also, it is called *clustering* when all the input samples $x \in X$ that bear the same label in y forms a connected subset in X.
- **Item 2.2** *Y consists of the k_r bits binary vectors $y = [y_1, \cdots, y_{k_r}]$, $y_j \in \{0, 1\}$.* That is, x is mapped into one binary code of k_r bits, which is called as *encoding*. The following two special cases are of particularly interesting:
  - **(a)** *Factorial encoding* if the bits of y are independent $p(y) = \prod_{i=1}^{k_r} p(y_i)$.
  - **(b)** *Exclusive encoding* if it is constrained that $\sum_{i=1}^{k_r} y_i = 1$ with $\sum_{i=1}^{k_r} p(y_i = 1) = 1$, which is actually the equivalent form of the above $y \in [1, 2, \cdots, k_r]$ and thus corresponds to *pattern recognition*.
- **Item 2.3** *Y consists of the k_r dimensional real vectors $y = [y_1, \cdots, y_{k_r}]$, $y_j \in R$.* In this case, usually X is also assumed to be real in R^d. That is, an d dimensional vector x is mapped into a $k_r (\leq d)$ dimensional real vector y, which is called *data transformation* in general. Also, it is called *data dimension reduction or feature extraction* when $k_r < d$. The following two special cases are of particularly interesting:
  - **(a)** *Independent Component Analysis (ICA).* In this case, it is assumed that each dimension of y is independent, i.e., $p(y) = \prod_{i=1}^{k_r} p(y_i)$. The well known *Principal Component Analysis (PCA)* is a special example in this case.
  - **(b)** *Distributed dimension reduction and Visualized map.* That is, x is mapped into one of q_r localized distributions in R^{k_r}, i.e., $p(y)$ is a finite mixture model or a mixture of gaussians. Particularly, when $k_r = 2$ or 3, we got an 2D or 3D visualized map for the distribution of the original high dimension data.

In all the cases above, the integer k_r represents the scale or complexity of representation, and its selection will be discussed later.

(2) **Architecture Design**.

We need to specify the architectures of four components $p_{M_x}(x)$, $p_{M_{y|x}}(y|x)$, $p_{M_{x|y}}(x|y)$ and $p_{M_y}(y)$, based on the given set of training samples together with some previous knowledge and assumption. For convenience, we let a denote one of elements in $\{x, x|y, y|x, y\}$ and S_a denote the architecture of $p_{M_a}(a)$. Roughly speaking, each $p_{M_a}(a)$ has three choices:

- **Item 2.4** *Fixed or Partially Fixed* . It means that $p_{M_a}(a)$ is simply fixed at some empirical estimation, based on a given set of training samples. In the case that the given set is only $D_x = \{x_i\}_{i=1}^N$ from an original density $p^o(x)$, i.e., for a pure unsupervised learning problem, $p_{M_x}(x)$ is fixed on some parametric or nonparametric empirical density estimation of the original $p^o(x)$, e.g., $p_{M_x}(x) = p_{h_x}(x)$ given by a nonparametric kernel estimate [4]:

$$p_{h_x}(x) = \frac{1}{\#D_x} \sum_{x_i \in D_x} K_{h_x}(x - x_i), \quad K_{h_x}(r) = \frac{1}{h_x^d} K(\frac{r}{h_x}), \tag{2.2}$$

  with a prefixed kernel function $K(.)$ and a prefixed smooth parameter h_x.
  In some practical cases, we may have a hybrid data set $D_H = \{D_{x,y}, D_x\}$ with $D_{x,y} = \{x_i, y_i\}_{i=1}^{N_2}$, $D_x = \{x_i\}_{i=1}^{N_1}$. That is, for some input x we know y, so we

can use not only $D_x = \{x_i\}_{i=1}^{N_1}$ to get $p_{h_x}(x)$ by eq.(2.2), but also $D_{x,y}$ to make $p_{M_{y|x}}(y|x)$ partially fixed at on some parametric or nonparametric empirical density estimation of the original $p^o(y|x)$, e.g., by the kernel estimates:

$$p_{h_x}(x) = \frac{1}{\#D_H} \sum_{x' \in D_H} K_{h_x}(x - x') = \frac{1}{\#D_H}[\sum_{x' \in D_x} + \sum_{(x',y) \in D_{x,y}}] K_{h_x}(x - x'), \quad (2.3)$$

$$p_{h_y}(y|x) = \begin{cases} \delta_d(y - y'), & (x, y') \in D_{x,y} \text{ (Item 2.1)}, \\ \frac{1}{h_y^{k_r}} K(\frac{y - y'}{h_y}), & (x, y') \in D_{x,y} \text{ (Item 2.2)} \end{cases}, \quad \delta_d(y) = \begin{cases} 1 & \text{for } y = 0; \\ 0, & \text{for } y \neq 0. \end{cases}$$

with a prefixed smooth parameter h_y. We call this learning problem as *semi-unsupervised learning* problem.

- **Item 2.5** *Free.* It means that $p_{M_a}(a), a \in \{x|y,\ y|x,\ y\}$ is a totally unspecified density or probability function in the form $p(a)$ without any constraint by itself. Thus, it is free to change such that it can be specified through other components indirectly. However, in order to have a useful Ying-Yang pair, its each part (i.e., either M_1 or M_2) should has *no more than one component free in the same time.* In this paper, we will simply use $p(a)$ to denote that $p_{M_a}(a)$ is free.
- **Item 2.6** *Parameterized Architecture.* It means that $p_{M_a}(a), a \in \{x|y,\ y|x,\ y\}$ is either a simple parametric density, e.g., a gaussian $p_{M_{x|y}}(x|y) = G(x, m_{x|y}, \Sigma_{x|y})$ with mean $m_{x|y}$ and variance matrix $\Sigma_{x|y}$, or a compounded parametric density with some of its parameters defined by a complicated function with a given parametric architecture consisting of a number elementary units that are organized in a given structure. Taking a three layer perceptron as an example, we have the following $S(x, W)$

$$S(x, W) = [m_{y_1}, \cdots, m_{y_{k_r}}], \quad m_{y_i} = s(\sum_{j=1}^{k^b} w_{i,j}^{(2)} s(x^T W_j^{(1)} + w_{j,0}^{(1)})), \quad (2.4)$$

as the mean $m_{y|x}$ of $p_{M_{y|x}}(y|x) = G(y, m_{y|x}, \sigma^2 I)$, where $W = \{w_{i,j}^{(2)}, W_j^{(1)}, w_{j,0}^{(1)}\}$, $s(u)$ is a sigmoid function, and k^b represents the scale or complexity of the parametric architecture.

Generally speaking, the design of a parameterized architecture consists of

- **(a)** Specification of density function form $p_a(a)$, e.g., we have $p_{M_{y|x}}(y|x) = G(y, S(x, W), \sigma^2 I)$ in eq.(2.4).
- **(b)** Specification of one or several types of elementary units in a fixed basic structure with a complexity k_a^b. For example, it can be a simple sigmoid neuron or a gaussian unit with k_a^b ignored, or it can be a m_{y_i} given by eq.(2.4) with a complexity k_a^b.
- **(c)** Specification of a structure on how to organize those elementary units into the architecture. For example, by the cascade organization of sigmoid neurons, we can get a three layer perceptron eq.(2.4).

(3) Model Selection

More precisely, it should be called as *Structural Scale Selection* for selecting the set of scale parameters $k = \{k_r, q_r, \{k_a^b\}\}$ with each elements defined as above. For the simple cases, some of the elements in k can disappear. Particularly, for eq.(2.2) we have $k_r = N$ and k_x^b is indirectly specified by the smooth parameter h_x. That is, $h_x = h(k_x^b)$ or $k_x^b = k_x^b(h_x)$. Actually, if we quantize h_x into a number of discrete values $h_1 < h_2 < \cdots h_{k_x^b} < \cdots$, we get a definite mapping between k_x^b and $h_{k_x^b}$.

(4) **Parameter Learning or Estimation**

After the above three levels of specifications, the only unspecified part for each component $p_{M_a}(a)$, $a \in \{x|y,\ y|x,\ y\}$ is a set θ_a of parameters in certain domains. Putting all of them together, we get the parameter set Θ to specify, which is usually done automatically by some optimization algorithm. In the literature, this task is also called *learning* simply in a narrow sense.

(5) **A Summary**

The learning in a BYY system is a process of the above four levels of specifications, based on the nature of the perception task, the given set of training samples, and some previous knowledge and assumption. This process consists of the following parts:

- **(a)** Based on the nature of the perception task, the domain Y and its complexity k_r are designed;
- **(b)** Based on the given set of training samples, $p_{M_x}(x)$ is prefixed at some non-parametric empirical estimate $p_{h_x}(x)$, e.g., by eq.(2.2) or eq.(2.3), and sometimes $p_{M_{y|x}}(y|x)$ is partially fixed by eq.(2.3);
- **(c)** Based on some previous knowledge, assumption and heuristics, architecture design is made on $S = \{S_{x|y}, S_{y|x}, S_y\}$ to specify whether each of them is free, and if not free, to further specify its parameterized architecture;
- **(d)** We also need to select structural scale $\{k^b_{x|y}, k^b_{y|x}, k^b_y\}$;
- **(e)** Then, we implement parameters learning on $\Theta = \{\theta_{x|y}, \theta_{y|x}, \theta_y\}$.

The entire process will be guided by the theory given in the next section.

3. Basic Bayesian Ying-Yang Learning Theory

Our basic theory is that the specifications of an entire Ying-Yang system should best enhance the so called *Ying-Yang Harmony or Marry*, through minimizing a harmony measure called *separation functional*:

$$\begin{aligned} F_s(M_1, M_2) &= F_s(p_{M_{y|x}}(y|x)p_{M_x}(x), p_{M_{x|y}}(x|y)p_{M_y}(y)) \geq 0, \\ F_s(M_1, M_2) &= 0, \ \textit{if and only if}\ p_{M_{y|x}}(y|x)p_{M_x}(x) = p_{M_{x|y}}(x|y)p_{M_y}(y), \end{aligned} \tag{3.1}$$

which describes the harmonic degree of the Ying-Yang pair. Such a learning theory is called as *Bayesian Ying-Yang (BYY) Learning Theory*. Particularly, if we only know D_x and the learning is based on prefixing $p_{M_x}(x)$ at some estimate from D_x, we call it *BYY Unsupervised Learning*; if we know a hybrid data set $D_H = \{D_{x,y}, D_x\}$ and the learning is based on prefixing $p_{M_x}(x)$ and prefixing $p_{M_{y|x}}(y|x)$ partially at some estimates from D_H, we call it *BYY Semi-unsupervised Learning*.

Generally speaking, $F_s(M_1, M_2)$ should increase as the discrepancy between $p_{M_{x|y}}(x|y)p_{M_y}(y)$ and $p_{M_{y|x}}(y|x)p_{M_x}(x)$ increases. Three categories of separation functionals, namely *Convex Divergence, L_p Divergence, and Decorrelation Index*, have been suggested in [14, 17]. Particularly, the *Convex Divergence* is defined as

$$F_s(M_1, M_2) = f(1) - \int_{x,y} p_{M_{y|x}}(y|x)p_{M_x}(x) f\left(\frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{p_{M_{y|x}}(y|x)p_{M_x}(x)}\right) dx\, dy, \tag{3.2}$$

where $f(u)$ is strictly convex on $(0, +\infty)$. When $f(u)$ is twice differentiable with $f(1) = 0$, eq.(3.2) is equivalent to Csiszar general divergence [1]. The BYY learning is called Bayesian Convex YING-YANG (BCYY) learning.

For convenience, in this whole paper we adopt such a convention that (a) $\int_u$ denotes the integral operation when u is known to be a real ; (b) in general when whether u is real or discrete is unclear yet, $\int_u$ denotes either the integral operation for a real u or the summation operation for a discrete u; (c) we explicitly use $\sum_u$ if u is already known to be discrete.

When $f(u) = u^\beta, 0 < \beta < 1$, eq.(3.2) becomes

$$F_s(M_1, M_2) = 1 - \int_{x,y} [p_{M_{y|x}}(y|x)p_{M_x}(x)]^{1-\beta}[p_{M_{x|y}}(x|y)p_{M_y}(y)]^\beta dx dy, \tag{3.3}$$

we call it as Positive Convex (PC) divergence, which is closely related to Renyi-α divergence. Interestingly, when $\beta = 0.5$, it leads to the Hellinger distance [1], which has a nice symmetric feature and can also be shown to be the only common special case of the above mentioned L_p Divergence and De-correlation Index [14, 17].

When $f(u) = \ln u$, eq.(3.2) reduces into the well known Kullback Divergence:

$$KL(M_1, M_2) = \int_{x,y} p_{M_{y|x}}(y|x)p_{M_x}(x) \ln \frac{p_{M_{y|x}}(y|x)p_{M_x}(x)}{p_{M_{x|y}}(x|y)p_{M_y}(y)} dx dy. \tag{3.4}$$

In this special case, the BYY learning is called Bayesian-Kullback YING-YANG (BKYY) learning.

Although the major function of a BYY system is determined by architecture design and a BYY system learned with different separation functionals under the same architecture design usually performs the same task, different separation functionals indeed bring some different features in learning and implementation.

The nice property $\ln xy = \ln x + \ln y$ makes the Kullback divergence the most elegant one for implementation. For example, we can decompose $KL(M_1, M_2)$ into

$$\begin{aligned}
KL(M_1, M_2) &= -H_{M_{y|x}} - Q(M_{y|x}, M_2) + C_{\{y, M_{1,2}\}} - H_{M_x}, \\
H_{M_{y|x}} &= \int_x p_{M_x}(x) H_{M_{y|x}}(x) dx, \quad H_{M_{y|x}}(x) = -\int_y p_{M_{y|x}}(y|x) \ln p_{M_{y|x}}(y|x) dy, \\
C_{\{y, M_{1,2}\}} &= -\int_y p_{M_1}(y) \ln p_{M_y}(y) dy, \quad p_{M_1}(y) = \int_x p_{M_{y|x}}(y|x) p_{M_x}(x) dx, \\
Q(M_{y|x}, M_2) &= \int_x p_{M_x}(x) Q_{\{M_{y|x}, M_{x|y}\}}(x) dx, \quad H_{M_x} = -\int_x p_{M_x}(x) \ln p_{M_x}(x) dx, \\
& \quad Q_{\{M_{y|x}, M_{x|y}\}}(x) = \int_y p_{M_{y|x}}(y|x) \ln p_{M_{x|y}}(x|y) dy.
\end{aligned} \tag{3.5}$$

As will be observed in the following sections, this decomposition brings us at least two advantages. First, it helps us to get further deep insights in the level of each or some combination of the four components. Second, the decomposition on the level of components usually will result in considerable saving in computation. In addition, from the next section, we can also observe that the property $\ln xy = \ln x + \ln y$ and the decomposition also help to best

exploit the structural features in the BYY systems with more complicated architectures.

Thus, BKYY learning is the most useful case and has been extensively studied in [12-23, 25-27]. In this paper, most cases will also concentrate on it and just consider some other separation functional at one specific occasion.

More formally, the theory given by eq.(3.1) provides theoretical guides on the following specific aspects:

– **Item 3.1 Parameter estimation or learning**, which is usually called *learning* in the narrow sense. That is, given S and k fixed, we determine

$$\Theta^* = arg \overset{g}{\min_{\Theta}} F_s(\Theta, S, k),$$
$$\text{"}arg\text{" means augment,} \qquad \text{"}\overset{g}{\min}\text{" denotes the global minimization.} \tag{3.6}$$

– **Item 3.2 Structural scale selection**, or called model selection. Given S, we determine

$$k^* = arg \overset{g}{\min_{k}} \mathcal{K}, \quad \mathcal{K} = \{j : J_1(j) = \overset{g}{\min_{k}} J_1(k)\}, \quad J_1(k) = F_s(\Theta^*, k, S). \tag{3.7}$$

That is, to pick the smallest one among those values of k that makes $J_1(k)$ reach its smallest value. In other words, we select the most economic structural scale when we have multiple choices. Usually, once $J_1(k)$ reaches its smallest value at k^*, it will keep this smallest value for all $k > k^*$.
We also have an alternative way for selecting k^*

$$k^* = arg \overset{g}{\min_{k}} J_2(k), \quad J_2(k) = -\int_{x,y} p_{M_1}(x,y)|_{\Theta^*} \ln p_{M_2}(x,y)|_{\Theta^*} dxdy, \tag{3.8}$$

where $p_{M_i}(x,y)|_{\Theta^*}, i = 1, 2$ denote the learned joint densities given in eq.(2.1) with the parameter Θ^* given by eq.(3.6). This $J_2(k)$ is a kind of complexity measure of the BYY system and is expected to be the smallest for the least complicated system. Usually, $J_2(k)$ reaches its smallest only at one value of k. In the most cases, the results of eq.(3.7) and eq.(3.8) are the same. However, each way has a different feature. For eq.(3.7), because $J_1(k)$ is actually random also, we should consider this issue when getting $\mathcal{K}$, especially for a limited number of samples. While for eq.(3.8), at some special cases, $J_2(k)$ may provide a wrong result because it biases to $k = 1$. In fact, $J_2(k)$ is just a part of $J_1(k)$.

– **Item 3.3 Architecture evaluation**. That is, for a set of architecture $\mathcal{S} = \{S^{(i)}, i = 1, \cdots, N_s\}$, we select the one $S^{(i^*)}$ with

$$i^* = \overset{g}{\min_{i}} J(S^{(i)}), \quad J(S^{(i)}) = -\int_{x,y} p_{M_1}(x,y)|_{\{\Theta^*,k^*\}} \ln p_{M_2}(x,y)|_{\{\Theta^*,k^*\}} dxdy, \tag{3.9}$$

where the meaning is similar to the case of eq.(3.8) above.

– **Item 3.4 Regularization**. For a limited number N of samples, regularization can be obtained via the following three ways:

- **(a)** Impose some structural constraint on one component to regularize the other components. For example, for a forward net we can design $S_{y|x}$ with more freedom to ensure its representation ability, but design $S_{x|y}$ with less freedom to regularize the learning to get a good generalization. Similarly, for a backward net or generative model, we can design $S_{x|y}$ with more freedom to ensure its representation ability, but design $S_{y|x}$ with less freedom to regularize the learning to get good generalization.
- **(b)** If we have several possibilities for estimating $p_{M_x}(x)$ and $p_{M_{y|x}}(y|x)$, we choose a best one that minimizes $F_s(M_1, M_2)$. For example, $p_{M_x}(x)$ and $p_{M_{y|x}}(y|x)$ are estimated by $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) and $p_{M_{y|x}}(y|x) = p_{h_y}(y|x)$ by eq.(2.3), the smooth parameters h_x, h_y can also be optimized by

$$\{h_x^*, h_y^*\} = arg \min_{\{h_x, h_y\}} F_s(\Theta^*, k^*, S^{(i^*)}, h_x, h_y). \tag{3.10}$$

- **(c)** Even interestingly, we can improve the generalization via *Re-sampling*. At the beginning, we can use some simple methods to get some estimates on $p_{M_x}(x)$ and $p_{M_{y|x}}(y|x)$ (e.g., to avoid the implementation difficulty of integral operations, we simply let $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) with $h_x = 0$ and $p_{M_{y|x}}(y|x) = p_{h_y}(y|x)$ by eq.(2.3) with $h_y = 0$ such that the integral operations are turned into summations). After learning, we can get new estimates on $p_{M_{y|x}}(y|x)$ and $p_{M_2}(x) = \int_y p_{M_{x|y}}(x|y)p_{M_y}(y)dy$. Then, we re-sample from them a number of new samples to add in the original training set, and use the enlarged training set to estimate $p_{M_x}(x)$ and $p_{M_{y|x}}(y|x)$ by the kernel estimates with $h_x = 0$ and $h_y = 0$ again. We can repeat the similar procedure if necessary until the resulted F_s reaches its most smallest value.

4. Structuralized BYY Systems and Theories

(1) The Architecture of YING-YANG System with Output Action for Supervised Learning. In a sister paper by the present author [12], the above basic BYY learning system and theory have been naturally extended into a more sophisticated form with three new components added for implementing various supervised classification and regression tasks such that not only the existing learning methods and theories for multilayer net, mixtures-of-experts, and radial basis function nets are unified as special cases with new learning algorithms, but also new selection criteria obtained for the number of hidden units and experts. Readers are referred to [12] for details.

(2) The Cascade or Ladder Architecture of YING-YANG Pairs. We consider the case that y consists of m subsets $y = \{y^{(j)}, j = 1, \cdots, m\}$ with each forming a layer that satisfies

Item 4.1 $y^{(j)}$ is only dependent to the immediate lower layer $y^{(j-1)}$ and immediate upper layer $y^{(j+1)}$, and $y^{(j-1)}$ and $y^{(j+1)}$ are independent under a given $y^{(j)}$. That is $p(y^{(j-1)}, y^{(j+1)}|y^{(j)}) = p(y^{(j-1)}|y^{(j)})p(y^{(j+1)}|y^{(j)})$, $p(y^{(j+1)}, y^{(j)}|y^{(j-1)}) = p(y^{(j)}|y^{(j-1)})$, and $p(y^{(j)}, y^{(j-1)}|y^{(j+1)}) = p(y^{(j)}|y^{(j+1)})$.

In this special case, we have

$$p_{M_y}(y) = p_{M_y}(y^{(m)}) \prod_{j=2}^{m} p_{M_y}(y^{(j-1)}|y^{(j)}), \quad p_{M_{y|x}}(y|x) = \prod_{j=1}^{m} p_{M_{y|x}}(y^{(j)}|y^{(j-1)}). \tag{4.1}$$

Moreover, we make the notations changes (a) $x^{(j)} = y^{(j-1)}$ and $p_{M_y}(y^{(j-1)}|y^{(j)}) = p_{M_{x|y}}(x^{(j)}|y^{(j)})$, (b) $x^{(1)} = x$ and $p_{M_{y|x}}(y^{(1)}|x^{(1)}) = p_{M_{y|x}}(y^{(1)}|x)$, and then substitute them into eq.(4.1), we further get

$$p_{M_y}(y) = p_{M_y}(y^{(m)}) \prod_{j=2}^{m} p_{M_{x|y}}(x^{(j)}|y^{(j)}), \quad p_{M_{y|x}}(y|x) = \prod_{j=1}^{m} p_{M_{y|x}}(y^{(j)}|x^{(j)}). \tag{4.2}$$

Next, we put them into eq.(3.4), we can get

$$\begin{aligned} KL(M_1, M_2) &= \sum_{j=1}^{m} KL_j(M_1, M_2), \\ KL_j(M_1, M_2) &= \int_{x^{(j)}, y^{(j)}} p_{M_{y|x}}(y^{(j)}|x^{(j)}) p_{M_{y|x}}(x^{(j)}|x^{(j-1)}) \\ &\times \ln \frac{p_{M_{y|x}}(y^{(j)}|x^{(j)}) p_{M_{y|x}}(x^{(j)}|x^{(j-1)})}{p_{M_{x|y}}(x^{(j)}|y^{(j)}) p_{M_y}(y^{(j)}|y^{(j+1)})} dx^{(j)} dy^{(j)}. \end{aligned} \tag{4.3}$$

Still, we make the notations changes $p_{M_x}(x^{(j)}) = p_{M_{y|x}}(x^{(j)}|x^{(j-1)})$ and $p_{M_y}(y^{(j)}) = p_{M_y}(y^{(j)}|y^{(j+1)})$, and put them into eq.(4.3), we finally get

$$KL_j(M_1, M_2) = \int_{x^{(j)}, y^{(j)}} p_{M_{y|x}}(y^{(j)}|x^{(j)}) p_{M_x}(x^{(j)}) \ln \frac{p_{M_{y|x}}(y^{(j)}|x^{(j)}) p_{M_x}(x^{(j)})}{p_{M_{x|y}}(x^{(j)}|y^{(j)}) p_{M_y}(y^{(j)})} dx^{(j)} dy^{(j)}, \tag{4.4}$$

Therefore, we see that the entire Ying-Yang system can be regarded as consisting of m Ying-Yang pairs in a cascade or ladder architecture. As a whole the entire system implements the following cascade bi-directional mappings:

$$\begin{aligned} x = x^{(1)} &\rightarrow y^{(1)} = x^{(2)} \rightarrow y^{(2)} = x^{(3)} \rightarrow \cdots \rightarrow x^{(m)} \rightarrow y^{(m)}, \\ y^{(m)} &\rightarrow x^{(m)} = y^{(m-1)} \rightarrow x^{(m-1)} = y^{(m-2)} \rightarrow \cdots \rightarrow y^{(1)} \rightarrow x^{(1)} = x. \end{aligned} \tag{4.5}$$

Thus, *the minimization of $KL(M_1, M_2)$ for the entire system is simply the minimization of the summation $\sum_{j=1}^{m} KL_j(M_1, M_2)$*. This kind of property makes us possible to concentrate on the learning of the basic form given in the previous section.

(3) The Star Architecture of YING-YANG Pairs. We consider the case that x consists of m subsets $x = \{x^{(j)}, j = 1, \cdots, m\}$ that satisfies

Item 4.2 The subsets $x^{(j)}$ becomes independent to each other under a given y. That is, $p(x^{(1)}, \cdots, x^{(m)}|y) = \prod_{j=1}^{m} p(x^{(j)}|y)$. In this special case, we have $p_{M_{x|y}}(x|y) = \prod_{j=1}^{m} p_{M_{x|y}}(x^{(j)}|y)$.

This property will simplify the term $Q(M_{y|x}, M_2)$ in eq.(3.5) into

$$Q(M_{y|x}, M_2) = \sum_{j=1}^{m} \int_{x_j} p_{M_x}(x_j) Q_{\{M_{y|x}, M_{x|y}\}}(x_j) dx_j,$$

$$Q_{\{M_{y|x},M_{x|y}\}}(x_j) = \int_y p_{M_{y|x}}(y|x_j)\ln p_{M_{x|y}}(x_j|y)dy \quad (4.6)$$

(4) The Tree Architecture of YING-YANG Pairs. By combining the above cascade and star architectures, we can obtain various tree architectures. In those cases, we can also exploit the architecture's features to facilitate the learning, in help of eq.(4.3), eq.(4.4), and eq.(4.6).

Finally, we should point out that for the cases that non-Kullback separation functionals are used, we cannot get the above nice features anymore. However, we can still heuristically enforce some of these features such that the learning on the above structuralized architectures can also be facilitated to some extent.

5. BKYY: Understandings and Implementations

In the case of Kullback divergence eq.(3.4), for a fixed $p_{M_x}(x)$ based on D_x, the term $-H_{M_x}$ in eq.(3.5) is irrelevant and thus can be ignored. That is, $\min_{M_1,M_2} KL(M_1,M_2)$ is equivalent to $\min_{\{M_{y|x},M_2\}} Kl(M_1,M_2)$ with

$$\begin{aligned}
Kl(M_1,M_2) &= \begin{cases} -H_{M_{y|x}} - Q(M_{y|x},M_2) + C_{\{y,M_{1,2}\}}, \\ \int_x p_{M_x}(x)KL_{y|x}(x)dx - L_{x,M_2}, \\ -Q(M_{y|x},M_2) + I_{M_{y|x}} + KL_y, \quad I_{M_{y|x}} = H_{M_{1,y}} - H_{M_{y|x}}, \end{cases} \\
KL_{y|x}(x) &= \int_y p_{M_{y|x}}(y|x)\ln\frac{p_{M_{y|x}}(y|x)}{p_{M_2}(y|x)}dy, \quad L_{x,M_2} = \int_x p_{M_x}(x)\ln p_{M_2}(x)dx, \\
p_{M_2}(y|x) &= \frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{p_{M_2}(x)}, \quad p_{M_2}(x) = \int_y p_{M_{x|y}}(x|y)p_{M_y}(y)dy, \\
H_{M_{1,y}} &= -\int_y p_{M_1}(y)\ln p_{M_1}(y)dy, \quad KL_y = \int_y p_{M_1}(y)\ln\frac{p_{M_1}(y)}{p_{M_y}(y)}dy. \qquad (5.1)
\end{aligned}$$

with the other notations kept the same as in eq.(3.5).

Here, we have three decompositions for $Kl(M_1,M_2)$. This first one is the most convenient one for implementation. The other two are useful in some specific cases. Moreover, they also provide us two other types of interesting interpretations for the BKYY learning as follows:

- **Item 5.1** $p_{M_x}(x)$ is an approximation of the true density $p^o(x)$, while $p_{M_2}(x)$ is the marginal density or called mixture density by the Ying model. Thus, L_{x,M_2} is the log-likelihood function of $p_{M_2}(x)$. Moreover, $p_{M_2}(y|x)$ can be regarded as the Ying model's mirror of $p_{M_{y|x}}(y|x)$. Therefore, the BKYY learning attempts to do maximum likelihood fitting on x with $p_{M_2}(x)$ and to minimize the expected discrepancy $KL_{y|x}(x)$ between the Yang passage and its mirror.
- **Item 5.2** $Q(M_{y|x},M_2)$ is an approximation of the expectation of the mixed log-likelihood $Q_{\{M_{y|x},M_{x|y}\}}(x)$, which represents the fit between a pattern generated via the Ying passage from y and the current input $x \in X$, and the fit is weighted or coordinated by the probability $p_{M_{y|x}}(y|x)$ of the y under the current x by the inverse mapping via the Yang passage. KL_y represents the discrepancy between $p_{M_y}(y)$(i.e., the original density in Y) and $p_{M_1}(y)$ (i.e., the density in Y described by the Yang model). Moreover, $I_{M_{y|x}}$ is the information transmitted via the Yang passage. Therefore, the BKYY learning is equivalent to minimize the weighted fit between the input pattern x and the pattern generated from the

Ying passage and to minimize the discrepancy between the original density in Y and the density in Y described by the Yang model, as well as to minimize the information transmitted via the Yang passage. The last minimization seems counter-intuitive at first glance. Actually, it is a regularization to avoid the over-fit of the Ying-Yang system on the current $p_{M_x}(x)$ through preventing the Yang passage from over-coordination.

Moreover, from eq.(5.1) we can also easily get the following two facts which will be used in the subsequent sections:

– **Item 5.3** When $p_{M_{y|x}}(y|x)$ is free, $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ will result in

$$p_{M_{y|x}}(y|x) = p_{M_2}(y|x) = p_{M_{x|y}}(x|y)p_{M_y}(y)/p_{M_2}(x) \tag{5.2}$$

This point can be easily observed from $KL_{y|x}(x) = 0$.

– **Item 5.4** When $p_{M_y}(y)$ is free, $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ will result in

$$p_{M_y}(y) = p_{M_1}(y) = \int_x p_{M_{y|x}}(y|x)p_{M_x}(x)dx. \tag{5.3}$$

This point can be easily observed from $KL_y = 0$.

In eq.(3.5) and eq.(5.1), to avoid the difficulty of computing integral in implementation, we approximately let $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) and $p_{M_{y|x}}(y|x) = p_{h_{y}}(y|x)$ by eq.(2.3) simply with $h_x = 0$ and $h_y = 0$, and obtain:

$$\begin{aligned}
&L^u_{x,M_2} = \frac{1}{\#D_x}\sum_{x\in D_x} \ln p_{M_2}(x), \quad L^{su}_{x,M_2} = \gamma_1 L^u_{x,M_2} + \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y)\in D_{x,y}} \ln p_{M_2}(x);\\
&Q^u(M_{y|x}, M_2) = \frac{1}{\#D_x}\sum_{x\in D_x} Q_{\{M_{y|x}, M_{x|y}\}}(x),\\
&Q^{su}(M_{y|x}, M_2) = \gamma_1 Q^u(M_{y|x}, M_2) + \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y)\in D_{x,y}} \ln p_{M_{x|y}}(x|y);\\
&H^u_{M_{y|x}} = \frac{1}{\#D_x}\sum_{x\in D_x} H_{M_{y|x}}(x), \quad H^{su}_{M_{y|x}} = \gamma_1 H^u_{M_{y|x}} - \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y)\in D_{x,y}} \ln p_{M_{y|x}}(y|x);\\
&p^u_{M_1}(y) = \frac{1}{\#D_x}\sum_{x\in D_x} p_{M_{y|x}}(y|x), \quad p^{su}_{M_1}(y) = \gamma_1 p^u_{M_1}(y) + \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y')\in D_{x,y}} \delta_d(y-y');\\
&\gamma_1 = \frac{\#D_x}{\#D_x + \#D_{x,y}}, \quad \gamma_2 = \frac{\#D_{x,y}}{\#D_x + \#D_{x,y}},
\end{aligned} \tag{5.4}$$

where the superscript "u" and "su" denote the cases of unsupervised and semi-unsupervised learning respectively. That is, it is equivalent to use the empirical averages to approximate the above mentioned expectations.

Correspondingly, we have that $\min_{M_1,M_2} KL(M_1, M_2)$ is equivalent to $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ with

$$Kl(M_1, M_2) = \begin{cases} \frac{1}{\#D_x}\sum_{x\in D_x} KL_{y|x}(x) - L^u_{x,M_2}, \\ -H^u_{M_{y|x}} - Q^u(M_{y|x}, M_2) + C_{\{y, M_{1,2}\}}, \end{cases} for\ unsupervised\ learning; \tag{5.5}$$

$$Kl(M_1, M_2) = \begin{cases} \frac{\gamma_1}{\#D_x}\sum_{x\in D_x} KL_{y|x}(x) + \frac{\gamma_2}{\#D_{x,y}}\sum_{(x,y)\in D_{x,y}} \ln \frac{p_{M_{y|x}}(y|x)}{p_{M_2}(y|x)} - L^{su}_{x,M_2}, \\ -H^{su}_{M_{y|x}} - Q^{su}(M_{y|x}, M_2) + C_{\{y, M_{1,2}\}}, \end{cases} \tag{5.6}$$

for semi-unsupervised learning.

In summary, $KL(M_1, M_2)$ given by eq.(3.4) or eq.(3.5) is the most general case for BKYY learning. $Kl(M_1, M_2)$ given by eq.(5.1) is a special case for a completely fixed $p_{M_x}(x)$ based on D_x such that the possibilities given by the previous Item 3.4(b)&(c) are omitted. $Kl(M_1, M_2)$ given by eq.(5.5) is a further specific case with $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) and $p_{M_{y|x}}(y|x) = p_{h_y}(y|x))$ by eq.(2.3) simply with $h_x = 0$ and $h_y = 0$.

As some indirect evidences for the justification of BKYY learning, in the following we show that several different specific expressions of $Kl(M_1, M_2)$ given by eq.(5.1) or eq.(5.5) have actually been used under different specific semantics and from different perspectives in the literature within the past decade already:

- **Item 5.5** In the special case of finite mixture with the specific design eq.(7.1) in Sec. 7, we can get that the $Kl(M_1, M_2)$ given by eq.(5.5) for the unsupervised case is actually equivalent to the cost function $D(W, \theta)$ proposed heuristically in [5], and also $\frac{1}{\#D_x}\sum_{x\in D_x} KL_{y|x}(x)$ is equivalent to $F(W, \theta)$ in [5] too.
- **Item 5.6** In the special case of gaussian mixture with same spherical variance, $Kl(M_1, M_2)$ in the above Item 5.5 at the special case, or equivalently $D(W, \theta)$, can be shown to be equivalent to the cost function F termed as *Helmholtz free energy* [6] derived from the minimum description length (MDL) principle for auto-encoders, if we ignore a constant part in F due to the quantization width.
- **Item 5.7** Denoting $p_{M_{x|y}}(x|y)p_{M_y}(y) = p(y, x|\theta)$. In the special case that $p_{M_{y|x}}(y|x) = p(y|x)$ is free, by combining the two terms $-Q^u(M_{y|x}, M_2) + C_{\{y,M_{1,2}\}}$ into one, we can see that the $Kl(M_1, M_2)$ given by eq.(5.5) for the unsupervised case is actually equivalent to the cost function $F(P, \theta)$ given by equation (5) in [10], derived heuristically but linked to " free energy" again.
- **Item 5.8** In the special case of finite mixture in general as well as gaussian mixture as an example, by ignoring some constants, the special case of $Kl(M_1, M_2)$ in the above Item 5.5, or equivalently $D(W, \theta)$, is also equivalent to a vector quantization cost function derived from minimizing the uncoded (i.e., error) bits and the wasted bits for the transmission channel's capacity [29].
- **Item 5.9** In the special case that y is a binary vector for representing all the hidden states of a multilayer networks with both bottom-up and top-down connection weights, $p_{M_{y|x}}(y|x)$ and $p_{M_{x|y}}(x|y)$ are two factorial (or factorizable within each layer) probability distributions corresponding to the both bottom-up and top-down connections, respectively, the instantaneous version of the $Kl(M_1, M_2)$ given by eq.(5.5) for the unsupervised case, i.e., $Kl(M_1, M_2)(x) = -H_{M_{y|x}}(x) - Q_{\{M_{y|x}, M_{x|y}\}}(x) + C_{\{y,M_{1,2}\}}$ can be shown to be equivalent to the free energy $F(d; \theta, Q)$ used in *Helmholtz Machine* [7, 2], obtained in the motivation of using it as a upper bound of the negative log-likelihood function of generating a data by the top-down connections as a generative model.

In this paper, $Kl(M_1, M_2)$ given by eq.(5.1) or eq.(5.5) is obtained as special cases of $KL(M_1, M_2)$ given by eq.(3.4), which is proposed as a harmony measure of a Ying-Yang pair in a BYY system for a unified general statistical learning framework. **Not only** it is interesting that

- **(a)** $Kl(M_1, M_2)$ from this new perspective coincides with those existing criteria in several specific cases above,
- **(b)** In comparison with $Kl(M_1, M_2)$, the general form $KL(M_1, M_2)$ is not a trivial generalization because the part contributed by $p_{M_x}(x)$ cannot be omitted for many cases, e.g, the case of the cascade architecture given in eq.(4.4) and the cases of improving the generalization as suggested by the previous Item 3.4(b)&(c);

But also from this new perspective we can obtain a number of new insights and a number of new uses just from $Kl(M_1, M_2)$ given by eq.(5.1) or eq.(5.5), as have been stated in Sec.3 and will be further elaborated later.

Before closing this section, we briefly discuss the *Convex Divergence* eq.(3.2), with which unfortunately we cannot get the decomposition similar to that in eq.(3.5). Therefore, $\min_{M_1,M_2} F_s(M_1, M_2)$ should be made as a whole. In its implementation, to avoid the difficulty of integral operation in implementation, we also approximately let $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) with $h_x = 0$ and $p_{M_{y|x}}(y|x) = p_{h_y}(y|x)$ by eq.(2.3) with $h_y = 0$. However, this time we can only do so for $p_{M_{y|x}}(y|x)p_{M_x}(x)$ outside $f(.)$; Otherwise, letting $h \to 0$ for the $p_{M_{y|x}}(y|x)p_{M_x}(x)$ inside $f(.)$ will result in a nonsense result. Basically, with the constant $f(1)$ ignored, we let

$$F_s^u(M_1, M_2) = -\frac{1}{\#D_x}\sum_{x\in D_x}\int_y p_{M_{y|x}}(y|x) f\left(\frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{p_{M_{y|x}}(y|x)p_h(x)}\right)dy, \tag{5.7}$$

$$F_s^{su}(M_1, M_2) = \gamma_1 F_s^u M_1, M_2 - \frac{\gamma_2}{\#D_{x,y}}\sum_{(x,y')\in D_{x,y}} \delta_d(y-y') f\left(\frac{p_{M_{x|y}}(x|y)p_{M_y}(y)}{p_{M_{y|x}}(y|x)p_h(x)}\right).$$

As will be discussed sometimes later, BCYY learning has some interesting features different from BKYY learning.

6. Features of Basic BYY Learning System and Theory

The above proposed Basic BYY learning system and theory have the following favorable features:

(1) Implemented by Alternative Minimization. The specific structure of the Ying-Yang system given in Fig.2.1 facilitates the implementation of $\min_{M_1,M_2} F_s$, which can be made by an iterative procedure to modify Θ, S, k in M_1, M_2 alternatively, namely we have

$$\begin{aligned} Step\quad 1\quad &:\quad Fix\ M_2 = M_2^{old},\ get\ M_1^{new} = arg\min_{M_1} F_s; \\ Step\quad 2\quad &:\quad Fix\ M_1 = M_1^{old},\ get\ M_2^{new} = arg\min_{M_2} F_s, \end{aligned} \tag{6.1}$$

which is guaranteed to reduce F_s until converged to one local minimum.

(2) A Unified Statistical Theory for Unsupervised and Semi-unsupervised Learning. Different designs on domain Y and on architecture $S = \{S_{x|y}, S_{y|x}, S_y\}$, as well as different choices on specific forms of separation functionals will lead us to quite a large number of different specific types of Ying-Yang system and thus a large number of specific learning models. Therefore, the BYY system with the above learning theory actually can function as a unified general statistical learning theory which can provide the specific systems and theories for the following major areas of unsupervised learning and its semi-unsupervised extension on the hybrid set D_H discussed in Item 2.4:

- **(a)** *BYY Pattern Recognition System and Theory.* It is the special case given by Item 2.1 or equivalently Item 2.2(b).

- **(b)** *BYY Factorial Encoding System and Theory.* It is the special case given by Item 2.2(a).
- **(c)** *BYY Data Dimension Reduction System and Theory.* It is the special case given by Item 2.3, including *Distributed dimension reduction and Visualized map*, which is a new data analysis tool.
- **(d)** *BYY Independent Component Analysis System and Theory*, which includes *Principal component analysis (PCA)* as a special case. It is the special case given by Item 2.3(a) and Item 2.2(a).

As will be shown in the subsequent sections, these BYY specific systems and theories will not only include the existing major learning models and theories in each area as special cases, but also provide them new theories for modeling or structural scale selection as well as architecture evaluation. Moreover, many new variants and new models can also be obtained with new interesting features.

(3) A New Framework for Improving Generalization in Finite Samples. For improving the generalization ability, the existing regularization approaches or generalization error's up-bound estimation methods (e.g., VC dimension method) introduce an extra penalty term to the original error cost function to be minimized together; and the existing Bayesian type approaches introduce a priori density on the parameters. Being different from all these existing approaches, the BYY learning system and theory holds the following interesting features for improving generalization:

- **(a)** The BYY learning system and theory introduce two complement architectures to regularize each other and use the selection criteria eq.(3.7) and eq.(3.9) to optimize the architectures such that the generalization can be improved.
- **(b)** Instead of targeting on minimizing the expected generalization error as an absolute standard by the existing approaches, the BYY learning system and theory aims at a relative standard — minimizing the discrepancy of the two learned complement models under the current set of finite samples.
- **(c)** The BYY system and theory only bases on the two complement Bayesian representations, there is no use of a priori on the parameters. Instead, a priori is embedded via the designs of the two complement architectures.
- **(d)** In addition, the two Bayesian representations may not be equal, i.e., the Bayesian rule may not be exactly true but only approximately holds. Therefore, our approach should not be confused with the existing Bayesian approach, but can be regarded a new type of *Structural and Relaxed* Bayesian approach.

(4) Being Easily Extended Into A Unified Statistical Theory For Various Structuralized Architectures for supervised learning as well as various structuralized learning purposes, as shown in Sec.4.

7. BYY Pattern Recognition System and Theory

(1) Finite Mixture, Pattern Recognition or Clustering Analysis

We consider the special case of Item 2.1 with $y = 1, \cdots, k_r$ and denote $k = k_r$. Moreover, we assume that $x \in R^d$ and $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2), with other architectures being $p_{M_{x|y}}(x|y) = p(x|\theta_y)$ and

$$p_{M_y}(y) = \alpha_y > 0, \ \sum_{y=1}^{k} \alpha_y = 1, \quad p_{M_{y|x}}(y|x) = p(y|x) \geq 0, \ \sum_{y=1}^{k} p(y|x) = 1. \tag{7.1}$$

That is, we have $p_{M_{x|y}}(x|y)$ is parametric, $p_{M_{y|x}}(y|x)$ and $p_{M_y}(y)$ are free probability functions. We also denote $\Theta_k = \{\alpha_y, \theta_y\}_{y=1}^k$.

As shown in [16], we can obtain a number of interesting results:

– **Item 7.1** Since $p_{M_{y|x}}(y|x)$ is free, from Item 5.3 and eq.(5.2) we have $p_{M_{y|x}}(y|x) = p_{M_2}(y|x)$ and $KL_{y|x}(x) = 0$. Thus, for a given k it follows also from eq.(5.4) that $\min_{\{p(y|x),\Theta_k\}} Kl(M_1, M_2)$ is equivalent to $\max_{\Theta_k} L(\Theta_k)$ with :

$$L(\Theta_k) = L^u_{x,M_2} = \frac{1}{\#D_x}\sum_{x\in D_x} \ln p(x,\Theta_k), \quad p(x,\Theta_k) = \sum_{y=1}^k \alpha_y p(x|\theta_y), \quad p(y|x) = \frac{p(x|\theta_y)\alpha_y}{p(x,\Theta_k)}, \tag{7.2}$$

where $L(\Theta_k)$ is the log-likelihood function of the finite mixture $p(x,\Theta_k)$. That is, *the parameter estimation part on Θ_k is equivalent to maximum likelihood learning of the finite mixture $p(x,\Theta_k)$*[3, 5].

– **Item 7.2** Since $p_{M_y}(y)$ is free, from Item 5.4 we have eq.(5.3). It further follows from eq.(5.4) that the Alternative Minimization algorithm eq.(6.1) becomes

$$\begin{aligned} &Step \quad 1 \quad : \quad From\ eq.(7.2),\ get\ p^{new}(y|x) = p(y|x)|_{\Theta_k=\Theta_k^{old}}; \\ &Step \quad 2 \quad : \quad \alpha_y^{new} = \frac{1}{\#D_x}\sum_{x\in D_x} p^{new}(y|x), \quad \theta_y^{new} = arg\max_{\theta_y}\{Q^u(M_{y|x}, M_2)\}; \end{aligned} \tag{7.3}$$

which is exactly the well known EM algorithm [3, 5, 24]. Here, we obtain it via a much simpler way with its convergence proved very easily.

– **Item 7.3** More interestingly, we can solve a long unsolved open problem—to select the best k^* by eq.(3.7) or eq.(3.8) with their specific forms as follows:

$$\begin{aligned} J_1(k) &= \frac{1}{\#D_x}\sum_{x\in D_x}\sum_{y=1}^k p^*(y|x)\ln p^*(y|x) + J_2(k), \\ J_2(k) &= -\frac{1}{\#D_x}\sum_{x\in D_x}\sum_{y=1}^k p^*(y|x)\ln p(x|\theta_y^*) - \sum_{y=1}^k \alpha_y^* \ln \alpha_y^*. \end{aligned} \tag{7.4}$$

where for each prefixed k, $p^*(y|x), \theta_y^*$ and α_y^* are the results of BYY learning eq.(7.2) by a given learning algorithm, e.g., the EM algorithm eq.(7.3).
Assume that k^o is the true number of the original mixture that x comes from. Under a very mild condition on $p(x|\theta_y)$ that can be satisfied by gaussian and other exponential family densities, we can prove that $J_1(k^o) < J_1(k)$ for $k < k^o$ and $J_1(k^o) = J_1(k)$ for $k \geq k^o$, as $Nh^d \to \infty$ and $h \to 0$[16]. Moreover, we can also prove $J_2(k^o) < J_2(k)$ for any $k \neq k^o$ for those cases that the densities in a mixture are not highly overlapped [14, 16].

– **Item 7.4** For the Gaussian mixture, i.e., $p(x|\theta_y) = G(x, m_y, \Sigma_y)$, the above EM algorithm eq.(7.3) will be simplified to the more specific form as given in [24]. More importantly, we can select the best number of gaussians by a further simplified form of $J_1(k)$ or $J_2(k)$ with [16, 20, 23, 25]:

$$J_2(k) = \sum_{y=1}^k \alpha_y^* \ln\sqrt{|\Sigma_y^*|} - \sum_{y=1}^k \alpha_y^* \ln \alpha_y^*. \tag{7.5}$$

– **Item 7.5** For the Gaussian mixture at the special case that $\Sigma_y = \sigma^2 I$ and $\alpha_y = 1/k$, using m_y as the center of a cluster and letting the corresponding $p(y|x)$ to be hard-cut into

$$I(y|x) = \begin{cases} 1, & \text{if } y = arg\ max_j p(j|x) = arg\ min_j ||x - m_j||^2, \\ 0, & \text{otherwise;} \end{cases} \tag{7.6}$$

then using $I(y|x)$ to replace $p(y|x)$ in the EM algorithm for Gaussian mixture in this special case, we will obtain exactly the well known k-means algorithm for the *Mean Square Error (MSE)* clustering analysis or vector quantization [16, 20, 23]. In the existing literature, however, k must be pre-given manually or heuristically [33]. More importantly, from eq.(7.5) we can get the following criterion for selecting the best number of k—a solution for an open problem unsolved for decades [16, 20, 23, 25]:

$$\begin{aligned} J_2^{g_h}(k) &= d \ln \sigma^* + \ln k, \quad or \quad J_2^{g_h}(k) = k \sqrt[k]{E_{MSE}^d}, \\ \sigma^2 &= \frac{E_{MSE}}{d \# D_x}, \quad E_{MSE} = \frac{1}{\# D_x} \sum_{x \in D_x} \sum_{y=1}^{k} I(y|x) ||x - m_y||^2. \end{aligned} \tag{7.7}$$

Moreover, the k-means algorithm for the MSE clustering is for the clusters of spherical shape with $\Sigma_y = \sigma^2 I$ only. For various special cases of $\Sigma_y \neq \sigma^2 I$, from the EM algorithm for Gaussian mixture we can obtain various types of extensions of the k-mean algorithm, including those so called Weighted MSE clustering, Mahalanobis distance clustering or elliptic clustering etc. Furthermore, we can also get various specific forms of the criterion eq.(7.5) for detecting the number of clusters [16, 23].

– **Item 7.6** More generally, we consider the finite mixture $p(x, \Theta_k)$ given by eq.(7.2) with $\sigma_{1,j}^2 \neq \sigma_{2,j}^2$ and $\theta_j = \{w_j, m_j, \sigma_{1,j}^2, \sigma_{2,j}^2\}$

$$p(x|\theta_j) = \frac{1}{\sqrt{2\pi}\sigma_{1,j}\sigma_{2,j}} e^{-0.5 d(x,\theta_j)}, \quad d(x,\theta_j) = \frac{||\frac{w_j^T(x-m_j)}{||w_j||}||^2}{\sigma_{1,j}^2} + \frac{||(I - \frac{w_j w_j^T}{||w_j||^2})(x - m_j)||^2}{\sigma_{2,j}^2}. \tag{7.8}$$

In this case, the EM algorithm eq.(7.3) has the following specific form:

$$\begin{aligned} &Step1: \quad From\ eq.(7.2),\ get\ p^{new}(j|x) = p(j|x)|_{\Theta_k = \Theta_k^{old}}; \\ &Step2: \quad \alpha_j^{new} = \frac{1}{\# D_x} \sum_{x \in D_x} p^{new}(j|x), \quad m_j^{new} = \frac{1}{\alpha_j^{new} \# D_x} \sum_{x \in D_x} p^{new}(j|x) x, \\ &\Sigma_j^{new} = \frac{1}{\alpha_j^{new} \# D_x} \sum_{x \in D_x} p^{new}(j|x)(x - m_j)(x - m_j)^T, \\ &w_j^{new}\ is\ a\ solution\ of\ eigen-equation\ \Sigma_j^{new} w_j^{new} = \lambda_j w_j^{new}, \\ &(\sigma_{1,j}^2)^{new} = \lambda_j, \ or\ (\sigma_{1,j}^2)^{new} = \frac{1}{\alpha_j^{new} \# D_x} \sum_{x \in D_x} p^{new}(j|x) ||\frac{w_j^T(x - m_j)}{||w_j||}||^2, \\ &(\sigma_{2,j}^2)^{new} = \frac{1}{\alpha_j^{new} \# D_x} \sum_{x \in D_x} p^{new}(j|x) ||(I - \frac{w_j w_j^T}{||w_j||^2})(x - m_j)||^2. \end{aligned} \tag{7.9}$$

where λ_j is either the largest or the smallest eigenvalue.

To have a deep insight on the result of this EM algorithm, we start to consider a simplest case of a single density $k = 1$. In this case, it is easy to know that m_j is the mean vector of D_x. Moreover, $w_j, \sigma_{1,j}^2, \sigma_{2,j}^2$ have two possible solutions as follows:

– (a) When λ_j is chosen as the largest eigenvalue in solving the above eigen-equation, $\sigma_{1,j}^2, w_j$ are the largest eigenvalue and the corresponding eigenvector of the covariance matrix of D_x, respectively, and $\sigma_{2,j}^2$ are the summation of all the remaining eigenvalues. That is, the learning is the so called *principal component analysis (PCA)* learning [11].

- (b) When λ_j is chosen as the smallest eigenvalue in solving the above eigen-equation, $\sigma_{1,j}^2, w_j$ are the smallest eigenvalue and the corresponding eigenvector of the covariance matrix of D_x, respectively, and $\sigma_{2,j}^2$ are the summation of all the remaining eigenvalues. That is, the learning is the so called *minor component analysis (MCA)* learning [35].

Then we consider the general case with $k > 1$. For each density localized at m_j, we have PCA or MCA, depending on whether its λ_j is chosen as the largest or the smallest eigenvalue. Thus, as a whole, we actually get the so called *localized PCA* or *localized MCA* or *localized hybrid PCA and MCA* learning [30, 31, 32].

Moreover, we can select the best number of densities by eq.(7.5) with $|\Sigma_y^*|$ replaced by the product $(\sigma_{1,j}^2)^{new*}(\sigma_{2,j}^2)^{new*}$.

Furthermore, we can modify Step 2 in eq.(7.9) into the following adaptive one:

$$\begin{aligned} Step2: \ & \alpha_j^{new} = (1-\gamma)\alpha_j^{old} + \gamma p^{new}(j|x), \quad m_j^{new} = (1-\gamma)m_j^{old} + \gamma \frac{p^{new}(j|x)}{\alpha_j^{new}} x, \\ & w_j^{new} = w_j^{old} - \gamma \frac{\partial d(x, w_j, m_j, \sigma_1, \sigma_2)}{\partial w_j}|_{w_j = w_j^{old}}, \\ & (\sigma_{1,j}^2)^{new} = (1-\gamma)(\sigma_{1,j}^2)^{old} + \gamma \frac{p^{new}(j|x)}{\alpha_j^{new}} \| \frac{w_j^T(x-m_j)}{\|w_j\|} \|^2, \\ & (\sigma_{2,j}^2)^{new} = (1-\gamma)(\sigma_{2,j}^2)^{old} + \gamma \frac{p^{new}(j|x)}{\alpha_j^{new}} \|(I - \frac{w_j w_j^T}{\|w_j\|^2})(x-m_j)\|^2, \end{aligned} \tag{7.10}$$

where $\gamma > 0$ is a learning stepsize. Thus, we get an adaptive algorithm, which has the same performance as the batch way EM algorithm eq.(7.9). Particularly, for a component density we can get PCA learning as long as $\sigma_{1,j}^2 > \sigma_{2,j}^2$ initially, and MCA as long as $\sigma_{2,j}^2 > \sigma_{1,j}^2$ initially.

In addition, in eq.(7.8), eq.(7.9) and eq.(7.11), the factor $\|w_j\|^2$ can be discarded under the constraint $\|w_j\|^2 = 1$.

- **Item 7.7** We can easily extend the above case into *localized principle subspace analysis (PSA)* or *localized minor subspace analysis (MSA)* as well as a hybrid of the two [30, 31, 32] by replacing the vector w_j with a matrix W_j which consists of q_j column vectors, as well as making the following changes:
  - (a) $\|w_j\|^2 = 1$ is replaced by $W_j^T W_j = I$, and the second choice of $d(x, \theta_j)$ in eq.(7.8) becomes $d(x, \Theta_j)$ with $\Theta_j = \{W_j, m_j, \sigma_{1,j}, \sigma_{2,j}\}$ and

$$d(x, \Theta_j) = \frac{\|(W_j^T W_j)^{-\frac{1}{2}} W_j^T (x-m_j)\|^2}{\sigma_{1,j}^2} + \frac{\|(I - W_j(W_j^T W_j)^{-1} W_j^T)(x-m_j)\|^2}{\sigma_{2,j}^2}. \tag{7.11}$$

  - (b) In Step 2 of eq.(7.9), W_j^{new} is a solution of eigen-equation $\Sigma_j^{new} W_j^{new} = W_j^{new} \Lambda_j$ with Λ_j consisting of the q_j largest eigenvalues of Σ_j^{new} for PSA or the q_j smallest eigenvalues of Σ_j^{new} for MSA.
  - (b) In Step 2 of eq.(7.9), the updating formulae for $(\sigma_{1,j}^2)^{new}$ and $(\sigma_{2,j}^2)^{new}$ should also be modified accordingly with eq.(7.11).

  Again, we can select the best number of densities by eq.(7.5) in the same way as in Item 7.6.
- **Item 7.8** All the previously discussed cases can be extended to their extensions with *semi-unsupervised learning*, what we need is the following substitutions:
  - (a) In eq.(7.2), we replace $L(\Theta_k) = L_{x,M_2}^u$ by $L(\Theta_k) = L_{x,M_2}^{su}$ from eq.(5.4);
  - (b) In eq.(7.3), we replace $Q^u(M_{y|x}, M_2)$ by $Q^{su}(M_{y|x}, M_2)$ from eq.(5.4);
  - (c) In eq.(7.3), we let α_y^{new} is given by

$$\alpha_y^{new} = \frac{\gamma_1}{\#D_x} \sum_{x \in D_x} p^{new}(y|x) + \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y') \in D_{x,y}} \delta_d(y-y'), \tag{7.12}$$

  and replace $Q^u(M_{y|x}, M_2)$ by $Q^{su}(M_{y|x}, M_2)$ from eq.(5.4);

– (d) The criterion eq.(7.4) is replaced by

$$\begin{aligned} J_1(k) &= \frac{\gamma_1}{\#D_x}\sum_{x\in D_x}\sum_{y=1}^{k} p^*(y|x)\ln p^*(y|x) + J_2(k), \\ J_2(k) &= -\sum_{y=1}^{k}\alpha_y^*\ln\alpha_y^* - \frac{\gamma_1}{\#D_x}\sum_{x\in D_x}\sum_{y=1}^{k} p^*(y|x)\ln p(x|\theta_y^*) \\ &- \frac{\gamma_2}{\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\sum_{y=1}^{k}\delta_d(y-y')\ln p(x|\theta_y), \end{aligned} \tag{7.13}$$

– (e) For gaussian mixture, the EM algorithm eq.(7.3) becomes:

$$\begin{aligned} &Step\quad 1\quad :\quad get\ p^{new}(y|x) = \frac{\alpha_y^{old}G(x,m_y^{old},\Sigma_y^{old})}{\sum_{y=1}^{k}\alpha_y^{old}G(x,m_y^{old},\Sigma_y^{old})}; \\ &Step\quad 2\quad :\quad get\ \alpha_y^{new}\ by\ eq.(7.12),\ and\ get \\ m_y^{new} &= \frac{\gamma_1}{\alpha_y^{new}\#D_x}\sum_{x\in D_x} p^{new}(y|x)x + \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\delta_d(y-y')x, \\ \Sigma_y^{new} &= \frac{\gamma_1}{\alpha_y^{new}\#D_x}\sum_{x\in D_x} p^{new}(y|x)(x-m_y)(x-m_y)^T \\ &+ \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\delta_d(y-y')(x-m_y)(x-m_y)^T. \end{aligned} \tag{7.14}$$

Again, we can select the best number of gaussians by eq.(7.5) with the converged parameters by the above algorithm.
Particularly, for the special case that $\Sigma_y = \sigma^2 I$ and $\alpha_y = 1/k$, via hard-cutting $p(y|x)$ into $I(y|x)$ by eq.(7.6) after the above Step 1 and then using $I(y|x)$ to replace all $p(y|x)$ in the Step 2, we can get a semi-unsupervised extension of the k-means algorithm. Moreover, in eq.(7.7), E_{MSE} is replaced by

$$E_{MSE} = \frac{\gamma_1}{\#D_x}\sum_{x\in D_x}\sum_{y=1}^{k} I(y|x)||x-m_y||^2 + \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\delta_d(y-y')||x-m_y||^2 \tag{7.15}$$

– (f) In the Step 2 of eq.(7.9), $\alpha_j^{new}, m_j^{new}, \Sigma_j^{new}$ are given by those in Step 2 of the above eq.(7.14). Moreover, we update

$$\begin{aligned} (\sigma_{1,j}^2)^{new} &= \frac{\gamma_1}{\alpha_j^{new}\#D_x}\sum_{x\in D_x} p^{new}(j|x)||\frac{w_j^T(x-m_j)}{||w_j||}||^2, \\ &+ \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\delta_d(y-y')||\frac{w_j^T(x-m_j)}{||w_j||}||^2, \\ (\sigma_{2,j}^2)^{new} &= \frac{\gamma_1}{\alpha_j^{new}\#D_x}\sum_{x\in D_x} p^{new}(j|x)||(I-\frac{w_jw_j^T}{||w_j||^2})(x-m_j)||^2 \\ &+ \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}}\delta_d(y-y')||(I-\frac{w_jw_j^T}{||w_j||^2})(x-m_j)||^2. \end{aligned} \tag{7.16}$$

In addition, in eq.(7.11), for $(x,y') \in D_{x,y}$ we just use $\delta_d(y-y')$ replace $p^{new}(y|x)$.

(2) A General BYY Pattern Recognition System and Theory

We further consider the case of *Exclusive Encoding* given in Item 2.2(b) with a binary vector $y = [y_1, \cdots, y_{k_r}]$, $y_j \in \{0, 1\}$, $\sum_{j=1}^{k_r} y_j = 1$, i.e., y can only take those k different values in which only one element is 1 while all the others are zero. In this case, we have

$$p_{M_y}(y_j = 1) = \pi_j, \ \sum_{j=1}^{k} \pi_j = 1, \ p_{M_{y|x}}(y_j = 1|x) = \pi_j(x), \ \sum_{j=1}^{k} \pi_j(x) = 1. \tag{7.17}$$

When $p_{M_y}(y)$ is free, from Item 5.4 we have eq.(5.3). It further follows from eq.(5.4) that

$$\pi_j = \frac{1}{\#D_x} \sum_{x \in D_x} \pi_j(x), \tag{7.18}$$

Furthermore, we consider a general design on the density $p_{M_{x|y}}(x|y)$ and the probability function $p_{M_{y|x}}(y|x)$ given by

$$p_{M_{x|y}}(x|y) = \prod_{j=1}^{k} p_j(x|y, \theta_j)^{y_j}, \quad \pi_j(x) = \frac{e^{g_j(x,\mu)}}{\sum_{j=1}^{k} e^{g_j(x,\mu)}}, \tag{7.19}$$

where $g_j(x, \mu), j = 1, \cdots, k$ are parametric functions with its architecture pre-designed. Two typical examples are $[g_1(x, \mu), \cdots, g_k(x, \mu)]$ being the output of (a) a multilayer perceptron and (b) a radial basis function networks. For $p_{M_{x|y}}(x|y)$, it is easy to observe that we have $p_{M_{x|y}}(x|y) = p_j(x|y, \theta_j)$ when $y_j = 1$ or equivalently the j-th element of y is 1 while all the others are zero. In general, $p_j(x|y, \theta_j)$ has the following typical choices:

$$p_j(x|y, \theta_j) = \begin{cases} p_j(x|y, \theta_j) = \begin{cases} G(x, f_j(y, W_j), \Sigma_j), & \text{for real } x, \\ \prod_{n=1}^{k} q_{j,n}(y)^{x_n}(1 - q_{j,n}(y))^{1-x_n}, & \text{for binary } x; \end{cases} \\ p(x|y, \theta) = \begin{cases} G(x, f(y, W), \Sigma), & \text{for real } x, \\ \prod_{n=1}^{k} q_n(y)^{x_n}(1 - q_n(y))^{1-x_n}, & \text{for binary } x; \end{cases} \\ p_j(x|\theta_j) = \begin{cases} G(x, m_j, \Sigma_j), & \text{for real } x, \\ \prod_{n=1}^{d} q_j^{x_n}(1 - q_j)^{1-x_n}, & \text{for binary } x; \end{cases} \end{cases} \tag{7.20}$$

where $x = [x_1, \cdots, x_d]$, $0 \le q_j \le 1$, and $q_{j,n}(y) = (1 + exp(-f_{j,n}(y, W_j))^{-1}$ with $f_j(y, W_j) = [f_{j,1}(y, W_j), \cdots, f_{j,d}(y, W_j)]$ being the output of the j-th backward network from $y \to x$ with parameter set W_j, which is implemented by either a multilayer perceptron or a radial basis function networks. Particularly, when $f_j(y, W_j) = f(y, W)$ and $q_{j,n}(y) = q_n(y)$, we actually have only one network.

With the above design, we get a general BYY pattern recognition system with its learning made by $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ with $Kl(M_1, M_2)$ given by eq.(5.1) or eq.(5.5), implemented by eq.(6.1) in general with either unsupervised learning or semi-unsupervised learning. After learning, this system recognizes x to the class $y_j = 1$ as long as $\pi_j(x) = \max_r \pi_r(x)$.

In the following, we consider several of its special cases:

- **Item 7.9** Given $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.2). When (a) $p_j(x|y,\theta_j) = p(x|\theta_j)$ with $p(x|\theta_j)$ from exponential family; (b) the function family $\mathcal{G}_j$ represented by $g_j(x,\mu)$ is large enough such that $\alpha_j p(x|\theta_j) \in \mathcal{G}_j$, we have that $p(y_j|x)$ is effectively free to be $\alpha_j p(x|\theta_j)/\sum_{j=1}^{k}\alpha_j p(x|\theta_j)$ by $\min_{\{M_{y|x},M_2\}} Kl(M_1,M_2)$. Thus, via the mapping $j = \sum_{q=0}^{k-1} 2^q y_q$ we can return back to exactly the finite mixture case given in eq.(7.1). For example, we have this case when $p(x|\theta_j) = G(x,m_j,\Sigma_j)$ and $\mathcal{G}_j$ is a family of all the quadratic functions of x.
- **Item 7.10** Given $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.2) still. When (a) $p_j(x|y,\theta_j) = p(x|\theta_j)$ but (b) $p(y_j|x)$ given by eq.(7.19) is in its general case, we cannot get Step 1 in eq.(7.3). Instead, it should be replaced by $\mu^{new} = arg\min_{\mu} Kl(M_1,M_2)$. However, via the mapping $j = \sum_{q=0}^{k-1} 2^q y_q$ we can still update M_2 by Step 2 in eq.(7.3). In this case, the Yang model does not totally follow but regularizes or restricts the Ying model. We call this case as *constrained finite mixture* model and the alternative minimization algorithm via the above modification on eq.(7.3) as the *constrained EM* algorithm. In this case, we can still use eq.(7.4) for selecting the best number k^* as long as $p^*(y|x)$ is given by eq.(7.19) with μ^*, θ_y^* and α_y^* being the converged results by the *constrained EM* algorithm.
- **Item 7.11** When (a) $p(y_j|x)$ given by eq.(7.19) is in its general case, (b) $p_j(x|y,\theta_j)$ is one of the other different choices given in eq.(7.20), we can get a number of different forward and backward Ying-Yang pairs. All of them can be trained by $\min_{\{M_{y|x},M_2\}} Kl(M_1,M_2)$ implemented by eq.(6.1) or a specific algorithm obtained from eq.(6.1), via either unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.2) or semi-unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.3). We can also select the best k^* by eq.(3.7) with their specific form similar to eq.(7.4) or its semi-unsupervised version eq.(7.13).

(3) Variants from Non-Kullback Separation Functionals

By replacing the Kullback divergence with other non-Kullback separation functionals, we can get the corresponding variants for all the previously introduced specific cases of the BYY PR system and learning theory. For example, for the *Convex Divergence* eq.(3.2), in general we can implement learning by eq.(6.1) with $F_s(M_1,M_2)$ given by eq.(5.7), for either unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.2) or semi-unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.3).

Particularly, for the finite mixture given in eq.(7.1), in order to simplify the computation, we simply force $p(y|x)$ to be given by eq.(7.2), resulting in

$$
\begin{aligned}
&Step\quad 1\quad :\quad From\ eq.(7.2),\ get\ p^{new}(y|x) = p(y|x)|_{\Theta_k=\Theta_k^{old}};\\
&Step\quad 2\quad :\quad get\ \alpha_y^{new}\ by\ eq.(7.12),\ and\ \theta_y^{new} = arg\max_{\theta_y}\\
&\{\frac{\gamma_1}{\#D_x}\sum_{x\in D_x} f(\frac{p(x,\Theta_k)}{p_h(x)}) + \frac{\gamma_2}{\alpha_y^{new}\#D_{x,y}}\sum_{(x,y')\in D_{x,y}} \delta_d(y-y')f(\frac{p(x,\Theta_k)}{p_h(x)})\},\\
&\quad or\quad get\ \theta_y^{new}\ by\ solving\ \sum_{x\in D_H} w(y,x)\frac{d\ln p(x|\theta_y)}{d\theta_j} = 0;
\end{aligned}
\tag{7.21}
$$

$$
w(y,x) = \begin{cases} f'(\frac{p(x,\Theta_k)}{p_h(x)})\frac{p(x,\Theta_k)}{p_h(x)}p^{new}(y|x), & x\in D_x,\\ f'(\frac{p(x,\Theta_k)}{p_h(x)})\frac{p(x,\Theta_k)}{p_h(x)}\delta_d(y-y'), & (x,y')\in D_{x,y}, \end{cases} \qquad f'(u) = \frac{df(u)}{du}.
$$

Here, the original weight $p^{new}(y|x)$ is reweighted into $w(y,x)$, we call the corresponding EM algorithm as the *Re-weighted EM (REM)* algorithm.

In the special case of gaussian mixture, the above Step 2 will take the following specific form:

$$m_y^{new} = \frac{1}{\alpha_j^{new} \# D_H} \sum_{x \in D_H} w(y,x)x, \quad \Sigma_y^{new} = \frac{1}{\alpha_j^{new} \# D_H} \sum_{x \in D_H} w(y,x)(x - m_y^{new})(x - m_y^{new})^T. \tag{7.22}$$

The algorithm given by eq.(7.22) or eq.(7.22) is for semi-unsupervised learning. It will reduce into the cases for unsupervised learning when $D_H = D_x$ with $\gamma_1 = 1, \gamma_2 = 0, D_{x,y} = \emptyset$.

8. BYY Factorial Encoding System and Theory

(1) A General BYY Factorial Encoding System and Theory

We consider the special case of Item 2.2 (a) with $y = [y_1, \cdots, y_{k_r}]$, $y_j \in \{0, 1\}$ and denote $k = k_r$. We still use $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) with $h \to 0$. Moreover, other architectural designs are made as follows:

$$\begin{aligned}
p_{M_y}(y) &= \prod_{j=1}^{k} \pi_j^{y_j}(1-\pi_j)^{1-y_j}, \quad 0 \le \pi_j \le 1, \\
p_{M_{y|x}}(y|x) &= \prod_{j=1}^{k} \pi_j(x,\mu)^{y_j}(1-\pi_j(x,\mu))^{1-y_j}, \\
p_{M_{x|y}}(x|y) &= \begin{cases} p(x|y,\theta), & \text{in general,} \\ G(x, f(y,W), \Sigma), & \text{gaussian } x, \\ \prod_{n=1}^{d} q_n(y,W)^{x_n}(1-q_n(y,W))^{1-x_n}, & \text{for binary } x; \end{cases} \\
\pi_j(x,\mu) &= s(g_j(x,\mu)), \quad q_n(y,W) = s(f^{(n)}(y,W)),
\end{aligned} \tag{8.1}$$

where $s(r)$ is a sigmoid function, e.g., $s(r) = 1/(1+e^{-r})$ or others with its range on $[0,1]$. $g(x,\mu) = [g_1(x,\mu), \cdots, g_k(x,\mu)]$ are the output of a forward network which can be either a multilayer perceptron or a radial basis function networks, and $f(y,W) = [f_1(y,W), \cdots, f_d(y,W)]$ are the output of a backward network which can be either a multilayer perceptron or a radial basis function networks.

With the above design, we get *a general BYY Factorial Encoding System* with its learning made by $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$, implemented by eq.(6.1) in general, via either unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.2) or semi-unsupervised learning with $p_{M_x}(x) = p_{h_x}(x)$ by eq.(2.3). After learning, this system transforms x into a factorial binary code y.

From computational point of view, the above general design for $p_{M_{x|y}}(x|y)$ cannot avoid the summation over all the values of y in its computing on $Q(M_{y|x}, M_2)$ because we cannot factorize $p_{M_{x|y}}(x|y)$ in the same way as $y = [y_1, \cdots, y_k]$. To reduce this computational load, we change $p_{M_{x|y}}(x|y)$ into the following specific design with $E(y|x) = [\pi_1(x,\mu), \cdots, \pi_k(x,\mu)]$:

$$p_{M_{x|y}}(x|E(y|x)) = \begin{cases} p(x|E(y|x),\theta), \\ G(x, f(E(y|x),W), \Sigma), \\ \prod_{n=1}^{d} q_n(E(y|x),W)^{x_n}(1-q_n(E(y|x),W))^{1-x_n}; \end{cases} \tag{8.2}$$

Such a design is still reasonable because $p_{M_{x|y}}(x|y)$ is still a density or probability function.

With the above design, by using θ to denote anyone of the three choices (i) θ, (ii) W, Σ, and (iii) W, we have that $Kl(M_1, M_2)$ given by eq.(5.5) becomes

$$
\begin{aligned}
J(k, \mu, \theta) &= \begin{cases} -H^u_{M_{y|x}} - Q^u(M_{y|x}, M_2) + C^u_{\{y, M_{1,2}\}}, \\ -H^{su}_{M_{y|x}} - Q^{su}(M_{y|x}, M_2) + C^{su}_{\{y, M_{1,2}\}}; \end{cases} \\
H^u_{M_{y|x}} &= -\frac{1}{\#D_x} \sum_{x \in D_x} \sum_{j=1}^{k} [s(g_j) \ln s(g_j) + (1 - s(g_j)) \ln (1 - s(g_j))], \quad g_j = g_j(x, \mu), \\
H^{su}_{M_{y|x}} &= \gamma_1 H^u_{M_{y|x}} - \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y) \in D_{x,y}} \sum_{j=1}^{k} [y_j \ln s(g_j) + (1 - y_j) \ln (1 - s(g_j))]; \\
Q(M_{y|x}, M_2) &= \begin{cases} \frac{1}{\#D_x} \sum_{x \in D_x} p_{M_{x|y}}(x|E(y|x)), & \text{unsupervised} \\ \frac{1}{\#D_H} \sum_{x \in D_H} p_{M_{x|y}}(x|E(y|x)), & \text{semi-unsupervised}; \end{cases} \\
C^u_{\{y, M_{1,2}\}} &= -\frac{1}{\#D_x} \sum_{x \in D_x} \sum_{j=1}^{k} [s(g_j(x, \mu)) \ln \pi_j + (1 - s(g_j(x, \mu))) \ln (1 - \pi_j); \\
C^{su}_{\{y, M_{1,2}\}} &= \gamma_1 C^u_{\{y, M_{1,2}\}} - \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y) \in D_{x,y}} \sum_{j=1}^{k} [y_j \ln \pi_j + (1 - y_j) \ln (1 - \pi_j)]. \qquad (8.3)
\end{aligned}
$$

When $p_{M_y}(y)$ is free, from Item 5.4 and eq.(5.4), we have

$$
\pi_j = \begin{cases} \frac{1}{\#D_x} \sum_{x \in D_x} \pi_j(x, \mu), & \text{unsupervised}, \\ \frac{\gamma_1}{\#D_x} \sum_{x \in D_x} \pi_j(x, \mu) + \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y) \in D_{x,y}} y_j \pi_j(x, \mu), & \text{semi-unsupervised}. \end{cases} \qquad (8.4)
$$

Therefore, under a prefixed k for the number of bits of y, we have that the algorithm eq.(6.1) takes the following specific forms:

$$
\begin{aligned}
&Step\ 1 \quad : \quad get\ \mu^{new}\ that\ minimizes\ or\ reduces\ (e.g.,\ by\ gradient\ descent)\ J(k, \mu, \theta) \\
&\qquad by\ eq.(8.4),\ and\ get\ \pi_j\ by\ eq.(8.4); \\
&Step\ 2 \quad : \quad get\ \theta^{new}\ that\ maximizes\ or\ increases\ (e.g.,\ by\ gradient\ descent) \\
&\qquad Q^u(M_{y|x}, M_2)\ or\ Q^{su}(M_{y|x}, M_2)\ by\ eq.(8.3). \qquad (8.5)
\end{aligned}
$$

Furthermore, we can also select the best k^* by eq.(3.7) or eq.(3.8), with

$$
\begin{aligned}
J_1(k) &= J(k, \mu^*, \theta^*), \quad \{\mu^*, \theta^*\} = arg \min^{g}_{\{\mu, \theta\}} J(k, \mu, \theta), \\
J_2(k) &= \begin{cases} -Q^u(M_{y|x}, M_2) + C^u_{\{y, M_{1,2}\}}, \\ -Q^{su}(M_{y|x}, M_2) + C^{su}_{\{y, M_{1,2}\}}; \end{cases} \quad at\ \{\mu^*, \theta^*\}. \qquad (8.6)
\end{aligned}
$$

This μ^*, θ^* can be estimated via the above algorithm eq.(8.5).

(2) Several Interesting Specific Cases

In the following, we consider several specific examples:

– **Item 8.1** We consider the case that $p_{M_y}(y)$ and $p_{M_{y|x}}(y|x)$ are given by eq.(8.1) with $p_{M_y}(y)$ free, and $p_{M_{x|y}}(x|E(y|x))$ is given by eq.(8.3) with $p_{M_{x|y}}(x|E(y|x)) = G(x, f(E(y|x), W), I)$. In this case, we have

$$
g(x, \mu) = \mu^T x, \quad \mu = [\mu_1^T, \cdots, \mu_k^T]^T, \quad f(y, \mu) = W^T E(y|x) = W^T S(\mu^T x),
$$

$$p_{M_{x|y}}(x|E(y|x)) = \frac{1}{\sqrt{2\pi}} e^{-\|x - W^T S(\mu^T x)\|^2}, \quad S(\mu^T x) = [s(g_1(x,\mu)), \cdots, s(g_k(x,\mu))],$$

$$Q^u(M_{y|x}, M_2) = -d^u(\mu, W) + const, \quad d^u(\mu, W) = \frac{1}{\#D_x} \sum_{x \in D_x} \|x - W^T S(\mu^T x)\|^2,$$

$$Q^{su}(M_{y|x}, M_2) = -d^{su}(\mu, W) + const, \quad d^{su}(\mu, W) = \frac{1}{\#D_H} \sum_{x \in D_H} \|x - W^T S(\mu^T x)\|^2;$$

$$H^u_{M_{y|x}} = -\frac{1}{\#D_x} \sum_{x \in D_x} \sum_{j=1}^{k} [s(\mu_j^T x) \ln s(\mu_j^T x) + (1 - s(\mu_j^T x)) \ln (1 - s(\mu_j^T x))],$$

$$H^{su}_{M_{y|x}} = \gamma_1 H^u_{M_{y|x}} - \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y) \in D_{x,y}} \sum_{j=1}^{k} [y_j \ln s(\mu_j^T x) + (1 - y_j) \ln (1 - s(\mu_j^T x))];$$

$$C^u_{\{y, M_{1,2}\}} = -\frac{1}{\#D_x} \sum_{x \in D_x} \sum_{j=1}^{k} [s(\mu_j^T x) \ln \pi_j + (1 - s(\mu_j^T x)) \ln (1 - \pi_j)];$$

$$C^{su}_{\{y, M_{1,2}\}} = \gamma_1 C^u_{\{y, M_{1,2}\}} - \frac{\gamma_2}{\#D_{x,y}} \sum_{(x,y) \in D_{x,y}} \sum_{j=1}^{k} [y_j \ln \pi_j + (1 - y_j) ln(1 - \pi_j)], \quad (8.7)$$

and we further have $J(k, \mu, \theta) = J(k, \mu, W) + const$ and

$$J(k, \mu, W) = \begin{cases} \frac{1}{\#D_x} \sum_{x \in D_x} J(x, \mu, W), \\ \frac{1}{\#D_H} \sum_{x \in D_H} J(x, \mu, W) \end{cases}, \qquad J(x, \mu, W) =$$

$$\sum_{j=1}^{k} [s(\mu_j^T x) \ln \frac{s(\mu_j^T x)}{\pi_j} + (1 - s(\mu_j^T x)) \ln \frac{1 - s(\mu_j^T x)}{1 - \pi_j}] + \|x - W^T S(\mu^T x)\|^2. \quad (8.8)$$

With prefixed k, the minimization of $J(k, \mu, W)$ can be implemented by a special case of the algorithm eq.(8.5), that is, we have

$$\begin{array}{lll} Step & 1 & : \quad Update\ \mu\ in\ one\ step\ by\ gradient\ descent\ on\ J(k, \mu, W), \\ Step & 2 & : \quad Fixed\ S(\mu^T x)\ and\ then\ update\ W\ by\ the\ least\ square\ or\ gradient \\ & & descent\ technique\ to\ reduce\ d^u(\mu, W)\ or\ d^{su}(\mu, W)\ by\ eq.(8.7). \end{array} \quad (8.9)$$

After this learning, We select the best k^* by eq.(3.7) or eq.(3.8), with $J_1(k) = J(k, \mu^*, W^*)$ given by eq.(8.8) directly and $J_2(k) = J(k, \mu^*, W^*)$ given by eq.(8.8) with $J(x, \mu, W)$ replaced by

$$J(x, \mu, W) = -\sum_{j=1}^{k} [s(\mu_j^T x) \ln \pi_j + (1 - s(\mu_j^T x)) \ln (1 - \pi_j)] + \|x - W^T S(\mu^T x)\|^2. \quad (8.10)$$

For getting a deep insight, we further consider the unsupervised case with $\mu = W^T$. In this case, $J(k, \mu, W)$ in eq.(8.8) becomes $J(k, W)$ with $J(x, \mu, W)$ replaced by

$$J(x, W) = \sum_{j=1}^{k} [s(w_j x) \ln \frac{s(w_j x)}{\pi_j} + (1 - s(w_j x)) \ln \frac{1 - s(w_j x)}{1 - \pi_j}] + \|x - W^T S(Wx)\|^2. \quad (8.11)$$

which can be still implemented by eq.(8.9) such that we update W in Step 1 with only the one at the front of $S(Wx)$ fixed at its old value, and then we update W again in Step 2.

Interestingly, we observe that the minimization of $J(k, W)$ consists of the minimization of $\sum_{x \in D_x} \|x - W^T S(Wx)\|^2$ as a part. While this part is exactly the

Least Mean Square Error Reconstruction (LMSER) for nonlinear one layer net proposed in [34] together with gradient descent algorithm given there also. This nonlinear LMSER learning is not only shown in [34] to be able to automatically break the symmetry in self-organization on data, but also applied to implement *Independent Component Analysis (ICA)* later by [9]. However, up to now it lacks a theoretical analysis to understand it better.
The above link between $J(k,W)$ and this LMSER learning provides us a new insight. As discussed previously, the minimization of $J(k,\mu,\theta)$ or equivalently $J(k,W)$ as a whole is for building a system which can encode x into a factorial code y with independent bits. In other word, the minimization of $\sum_{x\in D_x}||x - W^T S(Wx)||^2$ is for minimizing the square error between x and its reconstruction $W^T S(Wx)$ from the factorial code y of independent components.
Therefore, the minimization of the above $J(k,W)$ given by eq.(8.11) and its general form in eq.(8.8) can be called *Factorial Encoding LMSER* for unsupervised learning as well as its semi-unsupervised extension.

– **Item 8.2** The Factorial Encoding LMSER can be extended. With σ^2 as a parameter to be learned too, we let $p_{M_{x|y}}(x|E(y|x)) = G(x, S(W^T x), \sigma^2 I)$. In this case, the minimization of $J(k,\mu,\theta)$ with respect to σ^2 will result in

$$\sigma^2 = \begin{cases} \frac{1}{\#D_x}\sum_{x\in D_x}||x-\mu(Wx)||^2, \\ \frac{1}{\#D_H}\sum_{x\in D_H}||x-\mu S(Wx)||^2; \end{cases} \tag{8.12}$$

$$J(x,W,\mu) = 0.5\ln\sigma^2 + \sum_{j=1}^{k}[s(\mu_j^T x)\ln\frac{s(\mu_j^T x)}{\pi_j} + (1-s(\mu_j^T x))\ln\frac{1-s(\mu_j^T x)}{1-\pi_j}].$$

After this learning, we can get $J_1(k) = J(k,\mu^*,W^*)$ with the above $J(x,\mu,W)$ in eq.(8.8) directly and $J_2(k) = J(k,\mu^*,W^*)$ with $J(x,\mu,W)$ replaced by

$$J(x,\mu,W) = 0.5\ln\sigma^2 - \sum_{j=1}^{k}[s(\mu_j^T x)\ln\pi_j + (1-s(\mu_j^T x))\ln(1-\pi_j)]. \tag{8.13}$$

– **Item 8.3** We assume that $p_{M_y}(y)$ and $p_{M_{y|x}}(y|x)$ are given by eq.(8.2), and $p_{M_{x|y}}(x|E(y|x)) = \prod_{n=1}^{k} q_n(E(y|x),W)^{x_n}(1-q_n(E(y|x),W))^{1-x_n}$ to be factorial instead of Gaussian. We also have $g(x,\mu) = \mu^T x$, and $[q_1(E(y|x),W),\cdots,q_d(E(y|x),W)] = S(f(y,W)) = S(WE(y|x)) = S(WS(\mu^T x))$. With this design, we can get a special case of the above $J(k,\mu,\theta)$ in the form

$$\begin{aligned} J(k,\mu,\theta) &= J(\mu,W,k) + const, \quad J(k,\mu,W) = \begin{cases} \frac{1}{\#D_x}\sum_{x\in D_x} J(x,\mu,W,k), \\ \frac{1}{\#D_H}\sum_{x\in D_H} J(x,\mu,W,k), \end{cases} \\ J(x,\mu,W,k) &= \sum_{j=1}^{k}[s(\mu_j^T x)\ln\frac{s(\mu_j^T x)}{\pi_j} + (1-s(\mu_j^T x))\ln\frac{1-s(\mu_j^T x)}{1-\pi_j}] \\ &- \sum_{n=1}^{d}[x'_n\ln q_n(E(y|x),W) + (1-x'_n)\ln(1-q_n(E(y|x),W))]. \end{aligned} \tag{8.14}$$

Moreover, the second term can be rewritten into an equivalent form $\sum_{n=1}^{d}[x'_n\ln\frac{x'_n}{q_n(E(y|x),W)} + (1-x'_n)\ln\frac{1-x'_n}{1-q_n(E(y|x),W)}]$. Furthermore, if we let x'_n approximated by its mean Ex'_n, then under such an approximation, for the case of unsupervised learning on $J(k,\mu,\theta)$ we get that $J(\mu,W,k)$ given by eq.(8.14) becomes exactly the same as the $-\mathcal{F}(\theta,\phi)$ given in equation (3.11) in [2] for the Deterministic Helmholtz machine under the special case of one hidden layer !

– **Item 8.4** From the perspective of eq.(8.14) and the above general BYY Factorial Encoding system and theory, we can also get some new insights and several new variants for this deterministic one hidden layer Helmholtz machine:
(a) The learning directly based on eq.(8.14) without letting x'_n approximated by its mean Ex'_n provides a variant which is more reasonable because x'_n is known from the training set directly and also even more importantly is that the correlation between the input data x and the hidden layer $E(y|x)$ are accounted instead of only considered the correlation between Ex and $E(y|x)$— a weak point of the deterministic one hidden layer Helmholtz machine as pointed out by [2]. In addition, such a learning can be implemented by a simplified form of eq.(8.5) through just alternatively updating μ and W by gradient method, which is known to be guaranteed to converge. Furthermore, from eq.(8.14) this learning can be easily extended to the case of semi-unsupervised learning.
(b) According to the previous Item 3.3, we can select the best k^* by eq.(3.7) or eq.(3.8), with $J_1(k) = J(k, \mu^*, W^*)$ given by eq.(8.14) directly and $J_2(k) = J(k, \mu^*, W^*)$ given by eq.(8.14) with $J(x, \mu, W, k)$ replaced by

$$\begin{aligned} J(x, \mu, W, k) &= -\sum_{j=1}^{k}[s(\mu_j^T x)\ln \pi_j + (1 - s(\mu_j^T x))\ln(1 - \pi_j)] \\ &- \sum_{n=1}^{d}[x'_n \ln q_n(E(y|x), W) + (1 - x'_n)\ln(1 - q_n(E(y|x), W))]. \end{aligned} \quad (8.15)$$

This is an issue untouched in Helmholtz machine [2, 7], although it is obviously important.
(c) Other choices for implementing $g(x, \mu)$ and $f(y, W)$ can be considered to get different variants. For examples, (i) $g(x, \mu)$ by a forward multilayer net, and $f(y, W)$ by a RBF net, (ii) $g(x, \mu)$ by a RBF net, and $f(y, W)$ by a backward multilayer net, (iii) both $g(x, \mu)$ and $f(y, W)$ by RBF nets; (iv) both $g(x, \mu)$ and $f(y, W)$ by multilayer nets.

(3) The Cascade Architecture

We consider the case given by the previous Item 4.1. In this case, generally speaking, we can get the extension of the general BYY Factorial Encoding system and theory given in Sec.8(a). Particularly, we can also get the specific variants of the Factorial Encoding LMSER in the cascade architecture. Moreover, we further consider a case by adding in the following feature:

Item 8.5 The components within each layers $y^{(j)} = [y_1^{(j)}, \cdots, y_{k_j}^{(j)}]$ are independent. In this case, if we treat each layer with $KL(M_1^{(j)}, M_2^{(j)})$ given by eq.(4.4) in the same way as we did in Item 8.3 with x'_n approximated by its mean Ex'_n still, we can get that $KL(M_1, M_2)$ given by eq.(4.3) is equivalent to the $-\mathcal{F}(\theta, \phi)$ given in equation (3.11) in [2] for the Deterministic Helmholtz machine in the general case of m layers. As in the case of Item 8.4(a), we can also directly use each training sample without letting x'_n approximated by its mean Ex'_n.

Furthermore, according to Item 3.3, we can also investigate the number of hidden units in each layer via $KL(M_1, M_2)$ given by eq.(4.3) and $KL(M_1^{(j)}, M_2^{(j)})$ given by eq.(4.4).

9. BYY Data Dimension Reduction System and Theory

We consider the special case of Item 2.3 with $y = [y_1, \cdots, y_{k_r}], y_j \in R$ and denote $k = k_r$. We assume that y has a density of finite mixture:

$$p_{M_y}(y) = p(y|\xi) = \sum_{j=1}^{n_y} \alpha_j p(y|\xi_j), \quad \alpha_j > 0, \quad \sum_{j=1}^{n_y} \alpha_j = 1. \tag{9.1}$$

and y generates the current input $x \in R^d, d > k$, via the Ying passage that consists of $n_{x|y}$ linear or nonlinear channels $x = f(y, W_j) + e^{(j)}_{x|y}, j = 1, \cdots, n_{x|y}$, disturbed by noises $e^{(j)}_{x|y}$ from $p(e^{(j)}_{x|y}|\phi_j)$. That is, we have

$$p_{M_{x|y}}(x|y) = \sum_{j=1}^{n_{x|y}} \gamma_j p(x - f(y, W_j)|\phi_j), \quad \gamma_j > 0, \quad \sum_{j=1}^{n_{x|y}} \gamma_j = 1. \tag{9.2}$$

The purpose of the so called data dimension deduction is to invert x back to fit the original low dimension y, via the Yang passage that consists of $n_{y|x}$ channels $g(x, \mu_j), j = 1, \cdots, n_{y|x}$, described by a finite mixture:

$$p_{M_{y|x}}(y|x) = \sum_{j=1}^{n_{y|x}} \beta_j p(y|x, g(x, \mu_j), \psi_j), \quad \beta_j > 0, \quad \sum_{j=1}^{n_{y|x}} \beta_j = 1. \tag{9.3}$$

We consider the case of unsupervised learning only. With the above architecture design and let $p_{M_x}(x) = p_{h_x}(x)$ given by eq.(2.2) with $h \to 0$, we get a general BYY data dimension reduction system with its learning $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ by eq.(5.1) and eq.(5.5), which is implemented by eq.(6.1) in general for determining all the parameters

$$\Theta = \{\alpha_j, \beta_j, \gamma_j, \mu_j, \psi_j, W_j, \phi_j\}. \tag{9.4}$$

We can also select the best k^*—the dimension of the original y according to Item 3.2 with

$$\begin{aligned} J_1(k) &= J_1(\Theta^*, k, \mathcal{N}^*), \quad \{\Theta^*, \mathcal{N}^*\} = arg \min_{\{\Theta, \mathcal{N}\}}{}^{g} J_1(\Theta, k, \mathcal{N}), \\ & \quad J_1(\Theta, k, \mathcal{N}) = Kl(M_1, M_2), \ \mathcal{N} = \{n_{y|x}, n_{x|y}, \{n_y\}\}, \\ J_2(k) &= J_2(\Theta^*, k, \mathcal{N}^*), \quad J_2(\Theta, k, \mathcal{N}) = -\int_{x,y} p_{M_1}(x, y)|_{\Theta^*} \ln p_{M_2}(x, y)|_{\Theta^*} dxdy. \end{aligned} \tag{9.5}$$

After learning, we map x back to y either stochastically according to $p_{M_{y|x}}(y|x)$ given by eq.(9.3) or deterministically by taking

$$E(y|x) = \sum_{j=1}^{n_{y|x}} \beta_j E[p(y|x, g(x, \mu_j), \psi_j)], \tag{9.6}$$

while at the same time its inverse mapping $E(x|y) = \sum_{j=1}^{n_{x|y}} \gamma_j f(y, W_j)$ provides a reconstruction of x with noises filtered out.

This general BYY data dimension reduction system will have different specific cases and simplifications. Basically speaking, for eq.(9.3) and eq.(9.2) the simplest case can be that $n_{y|x} = 1$ and $n_{x|y} = 1$. Here, the differences of $n_{y|x}$ and $n_{x|y}$ will not affect the functions that the system can perform but indeed affect the performance in implementing the functions. These functions are mainly based on the value of n_y and the density form of $p(y|\xi_j)$.

The system can implement at least the following functions:

- **Item 9.1** In eq.(9.1), when $p(y|\xi_j) = G(y, m_y^{(j)}, \Sigma_y^{(j)})$, the system maps x into not only a low dimension data y via eq.(9.6), but also into one of $n_{y|x}$ clusters or gaussians $G(y, m_y^{(j)}, \Sigma_y^{(j)}), j = 1, \cdots, m$. That is, the tasks of the data dimension reduction and unsupervised classification are combined together. Particularly, when the dimension of y is 2 or 3, i.e., $k = 2$ or 3, we get a kind of 2D or 3D visualizations of the high dimensional data, which will be a very useful tool for interactive data analysis by human eyes although it has not been studied in the literature yet.
- **Item 9.2** If $p_{M_y}(y)$ is independent on its components, then we will get a special case of independent component analysis that will be studied in Sec.10.
- **Item 9.3** In the case that $n_y = n_{y|x} = n_{x|y} = 1$, and we have the case of the linear dimension reduction $y = \mu^T x + e_{y|x}$ under the assumption that x is generated from y by the linear transform $x = W^T y + e_{x|y}$ with $E(y) = 0$ and $E(xe_{x|y}^T) = 0$. From $y = \mu^T x + e_{y|x} = \mu^T(W^T y) + \mu^T e_{x|y} + e_{y|x}$, it is desired that

$$\mu^T W^T = I, \quad -\mu^T e_{x|y} = e_{y|x}. \tag{9.7}$$

With this as a starting point and also with the following designs by gaussian densities

$$\begin{aligned} p_{M_{y|x}}(y|x) &= G(e_{y|x}, 0, \Sigma_{y|x}) = G(y, \mu^T x, \Sigma_{y|x}), \\ p_{M_{x|y}}(x|y) &= G(e_{x|y}, 0, \Sigma_{x|y}) = G(x, W^T y, \Sigma_{x|y}), \\ p_{M_y}(y) &= G(y, 0, \Sigma_y), \quad \Sigma_y = \Lambda_y \ is\ diagonal, \end{aligned} \tag{9.8}$$

we get a general BYY linear data dimension reduction system and theory by $\min_{\{M_{y|x}, M_2\}} Kl(M_1, M_2)$ under the constraint eq.(9.7), with $Kl(M_1, M_2)$ given by eq.(5.1) or eq.(5.5). In this case, $-\mu^T e_{x|y} = e_{y|x}$ actually implies $\mu^T \Sigma_{x|y} \mu = \Sigma_{y|x}$ which can be inserted into $G(y, \mu^T x, \Sigma_{y|x})$ to reduce the parameter $\Sigma_{y|x}$ during the learning. Thus from eq.(5.1), we have

$$\begin{aligned} H_{M_{y|x}} &= H_{y|x} + const, \ Q_{\{y|x,x|y\}} = Q_{x,y} + const, \ C_{\{y,M_{1,2}\}} = C_y + const, \\ Q_{x,y} &= -0.5\{\ln|\Sigma_{x|y}| + \int_x p_{M_x}(x) Tr[\Sigma_{x|y}^{-1} \int_y G(y, \mu^T x, \Sigma_{y|x}) e_{x|y} e_{x|y}^T dy] dx\}, \\ C_y &= 0.5\{\ln|\Lambda_y| + \int_x p_{M_x}(x) Tr[\Lambda_y^{-1} \int_y G(y, \mu^T x, \Sigma_{y|x}) yy^T dy] dx\}, \\ C_y &= 0.5\{\ln|\Lambda_y| + \int_x p_{M_x}(x) Tr[\Lambda_y^{-1}(\Sigma_{y|x} + \mu^T x x^T \mu)] dx\} \\ &= 0.5\{\ln|\Lambda_y| + Tr[\Lambda_y^{-1} \mu^T (\Sigma_{x|y} + R_x)\mu]\}, \\ R_x &= \int_x p_{M_x}(x) x x^T dx, \quad H_{y|x} = 0.5\{k + \ln|\mu^T \Sigma_{x|y} \mu|\}, \\ Kl(M_1, M_2) &= J(\Theta, k), \quad J(\Theta, k) = -H_{y|x} - Q_{x,y} + C_y, \quad \Theta = \{W, \mu, \Sigma_{x|y}, \Lambda_y\}. \end{aligned} \tag{9.9}$$

The learning by $\min_\Theta J(\Theta, k)$ can again be implemented by a simplified form of eq.(6.1). After learning, we can also select the best k^* according to eq.(9.5) which is now simplified into

$$J_1(k) = J(\Theta^*, k), \quad \Theta^* = arg \min_{\Theta} J(\Theta, k), \quad J_2(k) = J_2(\Theta^*, k), \quad J_2(\Theta, k) = -Q_{x,y} + C_y. \tag{9.10}$$

In the rest of this section we will show that a special case of this general linear dimension reduction system and theory includes principal component analysis, minor component analysis as well as their combinations.

(2) A Linear Dimension Reduction System and Theory.

We continue the case of Item 9.3 by further constraining $\mu = W^T, WW^T = I_k, \Sigma_{x|y} = \sigma^2 I$. In this special case, from eq.(9.7) and eq.(9.9), we have

$$\begin{aligned}
& e_{y|x} = -We_{x|y}, \Sigma_{y|x} = W\sigma^2 I_d W^T = \sigma^2 I_k, E(e_{x|y}e_{x|y}^T) = \sigma^2 I_d, \\
& e_{x|y} = x - W^T y = x - W^T(Wx + e_{y|x}) = x - W^T Wx - W^T e_{y|x}, \\
& Tr[e_{x|y}e_{x|y}^T] = \|x - W^T Wx\|^2 + Tr[e_{y|x}e_{y|x}] - 2Tr[(I - W^T W)xe_{y|x}^T W], \\
& \int_x p_{M_x}(x)(\int_y G(e_{x|y}, 0, \Sigma_{y|x})xe_{y|x}^T dy = E(xe_{y|x}^T) = 0, \\
& \int_x p_{M_x}(x)Tr[\int_y G(e_{x|y}, 0, \Sigma_{y|x})Tr[e_{x|y}e_{x|y}^T]dy]dx = E\|x - W^T Wx\|^2 + k\sigma^2, \\
& E\|x - W^T Wx\|^2 = \int_x p_{M_x}(x)\|x - W^T Wx\|^2 dx, \\
& e_{x|y} = x - W^T Wx + W^T We_{x|y}, \quad x - W^T Wx = e_{x|y} - W^T We_{x|y}, \\
& E\|x - W^T Wx\|^2 = Tr[(I - W^T W)E(e_{x|y}e_{x|y}^T)(I - W^T W)^T] = (d-k)\sigma^2, \\
& Q_{x,y} = -0.5\{d\ln\sigma^2 + \sigma^{-2}\int_x p_{M_x}(x)Tr[\int_y G(e_{x|y}, 0, \Sigma_{y|x})Tr[e_{x|y}e_{x|y}^T]dy]dx\} \\
Q_{x,y} &= -0.5\{d\ln\sigma^2 + d\}, \quad H_{y|x} = 0.5\{k + k\ln\sigma^2\}, \\
C_y &= 0.5\{ln|\Lambda_y| + Tr[\Lambda_y^{-1}W(R_x + \sigma^2 I)W^T]\}.
\end{aligned} \tag{9.11}$$

Therefore $J(k, \Theta)$ given in eq.(9.9) is simplified into

$$J(k, W, \Lambda_y) = (d-k)\ln\sigma^2 - k + ln|\Lambda_y| + Tr[\Lambda_y^{-1}W(R_x + \sigma^2 I)W^T], \quad \sigma^2 = \frac{1}{d-k}E\|x - W^T Wx\|^2. \tag{9.12}$$

From eq.(9.12), we can get the following interesting results:

- **Item 9.4** We prefix $\Lambda_y = diag[\lambda_1, \cdots, \lambda_k]$ with $\lambda_1 \geq \cdots \geq \lambda_k > 0$ such that $u_j = \sigma^{*-2} - (\lambda_j^{-1} - \frac{\sum_{j=1}^k \lambda_j^{-1}}{d-k}) > 0$, $\sigma^{*2} = \frac{1}{d-k}\sum_{j=k+1}^n \lambda_j^x$, where $\lambda_1^x \geq \cdots \geq \lambda_d^x$ are the eigenvalues of R_x. In this case, it follows from $E\|x - W^T Wx\|^2 = Tr[R_x - WR_x W^T]$ that $\frac{\partial J(k,\Theta)}{\partial W} = -\frac{\partial Tr[UWR_x W^T]}{\partial W}$ with $U = diag[u_1, \cdots, u_k]$. Thus, $\min_{\{W, WW^T = I\}} J(k, W, \Lambda_y)$ is equivalent to $\max_{\{W, WW^T = I\}} Tr[UWR_x W^T]$. Following the results of [34] we get that the row vectors of W will be the first k principal components of R_x respectively. That is, we get the true k-PCA[34]. Moreover, if $\lambda_j = \lambda$, for all j such that $u_j = u > 0$, we have that the row vectors of W spans the same subspace spanned by the first k principal component vectors of R_x. That is, we get the so called principal subspace analysis (PSA) [11, 34].
- **Item 9.5** In the case of Item 9.4, the problem of how to decide the dimension k still remains an important open question in the literature without theoretical guide available yet. Here, we can select the best $k^* = arg\min_k^g J_1(k)$ by eq.(3.7) or $k^* = arg\min_k^g J_2(k)$ eq.(3.8) with

$$\begin{aligned}
J_1(k) &= -k\ln\sigma^{*2} - k + J_2(k), \quad J_2(k) = d\ln\sigma^{*2} + ln|\Lambda_y^*| + k, \\
& \{W^*, \Lambda_y^*\} = arg \min_{\{W, \Lambda_y\}}^g J(k, W, \Lambda_y),
\end{aligned}$$

$$\Lambda_y^* = W^*(R_x + \sigma^{*2} I)W^{*T}, \quad \sigma^{*2} = \frac{1}{d-k} E\|x - W^{*T} W^* x\|^2,$$

$$or \quad J_1(k) = (d-k)\ln \frac{\sum_{j=k+1}^{d} \lambda_j^x}{d-k} + \sum_{j=1}^{k} \ln(\lambda_j^x + \frac{\sum_{j=k+1}^{d} \lambda_j^x}{d-k});$$

$$J_2(k) = d \ln \frac{\sum_{j=k+1}^{d} \lambda_j^x}{d-k} + \sum_{j=1}^{k} \ln(\lambda_j^x + \frac{\sum_{j=k+1}^{d} \lambda_j^x}{d-k}) + k. \tag{9.13}$$

- **Item 9.6** In the case of Item 9.4, we let $\lambda_1 \geq \cdots \geq \lambda_k > 0$ such that $d_j < 0$ for all j. We can get the complementary part of PCA, that is, the row vectors of W will be the eigenvectors of R_x that correspond to the k smallest eigenvalues, respectively, which is called *minor component analysis(MCA)*[30, 31, 32]. Moreover, we can also get the complementary part of PSA, called MSA[30, 31, 32].
- **Item 9.7** Instead of prefixing Λ_y in the cases of Item 9.4 and Item 9.6, we implement $\min_{\{W,\Lambda_y\}} J(k, W, \Lambda_y)$. We can similarly get that the row vectors of W^* will be the k eigenvectors of R_x such that $Tr[UWR_xW^T]$ is maximized, which is equivalent to PCA in some special cases, and to MCA in some other special cases, as well as a combination of PCA and MCA.

10. BYY ICA System and Theory

(1) A General ICA System and Theory.

The situation is quite similar to that discussed in Sec.9(1) for the general data dimension deduction system and theory. The key different point here is that $y = [y_1, \cdots, y_{k_r}]$ with $k = k_r$ can be either real as Item 2.3 or binary as Item 2.2(a) such that y is from an independent density:

$$p_{M_y}(y) = p(y|\xi) = \prod_{j=1}^{k} p(y_j|\xi_j) \tag{10.1}$$

where $p(y_j|\xi_j)$ is a parametric model. More generally, it can be

$$p(y_j|\xi_j) = \sum_{r=1}^{n_{y,j}} \alpha_{r,j} p(y_j|\xi_{r,j}), \quad \alpha_{r,j} > 0, \ \sum_{r=1}^{n_{y,j}} \alpha_{r,j} = 1. \tag{10.2}$$

We regard that this factorial y generates a pattern to fit the current input x, via the Ying passage that is the same as that previously given in eq.(9.2). The purpose of the so called Independent Component Analysis (ICA) is to attempt to invert x back to this y of independent components, through the Yang passage given exactly the same as in eq.(9.3).

With the above architecture design, from eq.(5.1) we get a general BYY ICA system with its learning made by the theory of $\min_{\Theta} J(\Theta, k, \mathcal{N})$ with $\Theta = \{\alpha_{r,j}, \beta_j, \gamma_j, \mu_j, \psi_j, \xi_j, \phi_j\}$ and

$$J(\Theta, k, \mathcal{N}) = -H_{y|x} - Q_{\{x,y\}} + C_y, \quad \mathcal{N} = \{n_{y|x}, n_{x|y}, \{n_{y,j}\}\}, \tag{10.3}$$

$$H_{y|x} = -\int_x p_{M_x}(x)[\int_y \sum_{j=1}^{n_{y|x}} \beta_j p(y|x, g(x, \mu_j)) \ln \sum_{j=1}^{n_{y|x}} \beta_j p(y|x, g(x, \mu_j)) dy] dx,$$

$$Q_{\{x,y\}} = \int_x p_{M_x}(x)[\int_y p_{M_{y|x}}(y|x)\ln\sum_{j=1}^{n_{x|y}}\gamma_j p(x-f(y,W_j)|\phi_j)dy]dx,$$

$$C_y = -\int_y p_{M_1}(y)\ln\prod_{j=1}^{k}p(y_j|\xi_j)dy, \quad p_{M_1}(y)=\sum_{j=1}^{n_{y|x}}\beta_j\int_x p(y|x,g(x,\mu_j))p_{M_x}(x)dx.$$

This $\min_\Theta J(\Theta,k,\mathcal{N})$ can be implemented by eq.(6.1) in general. We can also select the best k^*—the number of independent component by eq.(3.7) or eq.(3.8) with

$$\begin{aligned} J_1(k) &= J(\Theta^*,k,\mathcal{N}^*), \quad \{\Theta^*,\mathcal{N}^*\}=arg\min_{\{\Theta,\mathcal{N}\}}{}^{g} J(\Theta,k,\mathcal{N}), \\ J_2(k) &= J_2(\Theta^*,k,\mathcal{N}^*), \quad J_2(\Theta,k,\mathcal{N})=-Q_{\{x,y\}}+C_y. \end{aligned} \tag{10.4}$$

After learning, similar to Sec.9(1), we can also map x back to y and get a reconstruction of x with noises filtered out.

The above proposed form is the general one that covers all the possible cases. First, the generating channel as shown in eq.(9.2) includes the cases:

- **(a)** x is generated from y by either linear $f(y,W_j)=W_j^T y$ or nonlinear $f(y,W_j)$ channel and by either single or multiple channels.
- **(b)** x is generated from y either with noise or without noise and either gaussian noise or other noises.
- **(c)** the dimension d of x is either equal to $(d=k)$ or unequal to $(d\neq k)$ the dimension k of y, and the dimension k of y is either known or unknown.
- **(d)** the source y is either binary or real.

Second, the channel from x back to y as shown in eq.(9.3) includes the cases:

- **(e)** the inverting channel consists of either single or multiple models.
- **(f)** $p(y|x,g(x,\mu_j))$ is either gaussian or non-gaussian. When $p(y|x,g(x,\mu_j))$ is gaussian and the regression is $\int_y yp(y|x,g(x,\mu_j))dy=g(x,\mu_j)$, we have that $g(x,\mu_j)$ is either liner or nonlinear.

This general form can be further simplified into various specific forms with different levels of complexity according to different specific assumptions. The following are some examples:

- **Item 10.1** We consider the case that (a) $n_{y|x}=1$, $n_{x|y}=1$, and $d=k$, (b) $p(y_j|\xi_j)$ is given by eq.(10.1), (c) there is no noise, i.e., for infinite small volumes $V_y\to 0$ and $V_x\to 0$, we have

$$p_{M_{y|x}}(y|x)=\begin{cases}1/\delta V_y, & \text{if } y=g(x,\mu),\\ 0, & \text{otherwise;}\end{cases} \qquad p_{M_{x|y}}(x|y)=\begin{cases}1/\delta V_x, & \text{if } x=f(y,W),\\ 0, & \text{otherwise.}\end{cases} \tag{10.5}$$

From eq.(3.4) directly, we can simplify $J(\Theta,k,\mathcal{N})$ given in eq.(10.3) into

$$J(\mu,\xi)=\ln\frac{\delta V_x}{\delta V_y}-\sum_{j=1}^{k}\int_x p_{M_x}(x)\ln p(g_j(x,\mu)|\xi_j)dx, \quad where\ g(x,\mu)=[g_1(x,\mu),\cdots g_k(x,\mu)], \tag{10.6}$$

Particularly, when $g(x,\mu)=\mu^T x$ (i.e. the linear ICA model). We have $\ln\frac{\delta V_x}{\delta V_y}=-\ln|\mu|$ and eq.(10.6) becomes

$$J(\mu,\xi)=-\ln|\mu|-\sum_{j=1}^{k}\int_x p_{M_x}(x)\ln p(\mu_j^T x|\xi_j)dx, \quad where\ \mu_j\ is\ j-th\ column\ of\ \mu. \tag{10.7}$$

This is just the so called maximum likelihood ICA model [8], which is also shown to be equivalent to the mutual information method (MMI) by Amari, Cichocki, & Yang (1995) and Informax by Bell & Sejnowski (1995), both can be found in [8]. Particularly, when $p(y_j|\xi_j)$ is given by eq.(10.2), we get the finite mixture based implementation for the information theoretic ICA approach [17, 18, 19].

– **Item 10.2** All the others are kept the same as in Item 10.1, except we allow that the pre-given k can be smaller than d, i.e. $k \leq d$, and also noise is considered , i.e., $x = f(y, W) + e_x$ with $p_{M_{x|y}}(x|y) = G(x, f(y, W), \sigma^2 I)$ and $Eye_x^T = 0$. In this case, since $arg\min_{\mu,\xi,k}\{\lim_{V_y \to 0} H_{y|x}\} = \lim_{V_y \to 0}\{arg\min_{\mu,\xi,k} H_{y|x}\}$, we have $arg\min_{\mu,\xi,k} J(\mu, \xi, k) = arg\min_{\mu,\xi,k}[J(\mu, \xi, k) + H_{y|x}]$ and thus can ignore $H_{y|x}$. So, we get

$$\begin{aligned} Q_{\{y|x,x|y\}} &= -0.5\{\ln 2\pi + d\ln\sigma^2 + d\}, \quad \sigma^2 = d^{-1}E||x - f(g(x,\mu), W)||^2, \quad (10.8) \\ J(\mu, \xi, \sigma^2, k) &= 0.5d\ln E||x - f(g(x,\mu), W)||^2 - \sum_{j=1}^{k}\int_x p_{M_x}(x)\ln p(g_j(x,\mu)|\xi_j)dx. \end{aligned}$$

as a new nonlinear ICA model. Particularly, when $g(x, \mu) = \mu^T x, f(y, W) = W^T y,\ \mu = W^T$, we have a new linear ICA model as follows:

$$J(\mu, \xi, \sigma^2, k) = 0.5d\ln E||x - \mu\mu^T x||^2 - \sum_{j=1}^{k}\int_x p_{M_x}(x)\ln p(\mu_j^T x|\xi_j)dx. \quad (10.9)$$

For both the linear and nonlinear cases above, we can also select the best number of sources by $k^* = arg\min_k J(k)$ with $J(k) = \min_{\{\mu,\xi,\sigma^2\}}^{g} J(\mu, \xi, \sigma^2, k)$.

– **Item 10.3** In the case of Item 10.2, if we let $p_{M_{y|x}}(y|x) = G(y, g(x, \mu), \Sigma_{y|x})$ instead, then we have

$$\begin{aligned} H_{M_{y|x}} &= 0.5(k + \ln 2\pi + \ln|\Sigma_{y|x}|), \quad \Sigma_{y|x} = E[(y - g(x,\mu))(y - g(x,\mu))^T], \\ & \quad Q_{\{y|x,x|y\}} = -0.5\{\ln 2\pi + d\ln\sigma^2 + \frac{Q_{x,y}}{\sigma^2}\}, \\ Q_{x,y} &= \int_x p_{M_x}(x)[\int_y G(y, g(x,\mu), \Sigma_{y|x})||x - f(y, W)||^2 dy]dx, \\ C_y &= -\sum_{j=1}^{k}\int_x p_{M_x}(x)[\int_{y_j} G(y_j, g_j(x,\mu), \sigma^2_{j,y|x})\ln p(y_j|\xi_j)dy_j]dx, \\ J(\Theta, k) &= 0.5\{d\ln\sigma^2 - k - \ln|\Sigma_{y|x}| + \frac{Q_{x,y}}{\sigma^2}\} + C_y, \quad \Theta = \{\mu, \xi, \sigma^2, \Sigma_{y|x}\}. \quad (10.10) \end{aligned}$$

as a generalization of eq.(10.8) for nonlinear ICA, where $\sigma^2_{j,y|x}$ is the j-th diagonal element of $\Sigma_{y|x}$. When $g(x, \mu) = \mu^T x, f(y, W) = W^T y, \mu = W^T$, we have

$$\Sigma_{y|x} = E[(y-\mu^T x)(y-\mu^T x)^T], \quad Q_{x,y} = \int_x p_{M_x}(x)[\int_y G(y, g(x,\mu), \Sigma_{y|x})||x - W^T y||^2 dy]dx, \quad (10.11)$$

and use them to replace $\Sigma_{y|x}$ and $Q_{x,y}$ in eq.(10.10), we get a generalization of eq.(10.9) for linear ICA.

For both the linear and nonlinear cases above, we can also select the best number k^* of sources by eq.(3.7) or eq.(3.8) with

$$J_1(k) = \min_{\Theta}^{g} J(\Theta, k), \quad J_2(k) = \min_{\Theta}^{g} J_2(\Theta, k), \quad J_2(\Theta, k) = 0.5\{d\ln\sigma^2 + \frac{Q_{x,y}}{\sigma^2}\} + C_y. \quad (10.12)$$

- **Item 10.4** Being different from Item 10.3, in the case of Item 10.2 we let $p_{M_{y|x}}(y|x)$ given by eq.(8.1) and $p_{M_{x|y}}(x|y)$ given by eq.(8.2), then we again get the *Factorial Encoding LMSER learning* discussed in Item 8.1 and Item 8.2, which can also be used for linear and nonlinear ICA with best number of sources k selected under the situation that there is noise.
- **Item 10.5** We can also extend the cases discussed in Items 10.2, 10.3 and Item 10.4 by using $p(y_j|\xi_j)$ given by eq.(10.2), such that we can get even better performances based on using finite mixtures as did in [17, 18, 19].

11. Conclusions

Bayesian Ying-Yang (BYY) system and theory has been systematically introduced as a unified statistical learning approach on *parameter learning, regularization, structural scale selection, architecture designing and data sampling*. For unsupervised learning and its semi-unsupervised extension, this paper has shown how the general theory provides new theories for *unsupervised pattern recognition and clustering analysis, factorial encoding, data dimension reduction, and independent component analysis*, such that not only several existing popular unsupervised learning approaches are unified as special cases with new insights and new results, but also a number of new models and new results are obtained. In the sister papers [12] and [13], this theory has further been shown to function as a general theory for various problems of supervised learning and time series learning as well.

References

1. Csiszar, I. & Tusnady, G., " Information geometry and alternating minimization procedures", *Statistics and Decisions*, Supplementary Issue, No.1, pp205-237(1984).
2. Dayan, P., Hinton, G. E., Neal, R. N. & Zemel, R.S., " The Helmholtz machine", *Neural Computation* 7, No.5, 889-904 (1995).
3. Dempster, A., Laird, N. M., & Rubin, D. B. "Maximum-likelihood from incomplete data via the EM algorithm", *Journal of the Royal Statistical Society, B*, **39** (1), 1-38, (1977).
4. Devroye, L, *A Course in Density Estimation*, Birhhauser Publisher, Boston,.(1987).
5. Hathaway, R.J, "Another interpretation of the EM algorithm for mixture distributions", *Statistics & Probability Letters 4*, 53-56.(1986).
6. Hinton, G. E. & Zemel, R.S., "Autoencoders, minimum description length and Helmholtz free energy", *Advances in Neural Information Processing Systems 6*, eds, J.K.Cowan, G.Tesauro and J.Alspector, Morgan Kaufmann Pub: San Mateo, CA, 3-10(1994).
7. Hinton, G. E., Dayan, P., Frey, B.J. & Neal, R. N., "The wake-sleep algorithm for unsupervised learning neural networks", *Science 268*, 1158-1160(1995).
8. Jutten, C., "From source separation to Independent component analysis: An introduction to special session", Invited special session on Blind Signal Separation, Proc. of 1997 European Symp. on Artificial Neural Networks, Bruges, April 16-18, 243-248(1997).
9. Karhunen, J. & Joutsensalo, J.J., "Representation and separation of signals using nonlinear PCA type Learning," *Neural Networks* 7, 113-127 (1994).

10. Neal, R. N.& Hinton, G. E, "A new view of the EM algorithm that justifies incremental and other variants", CS Dept., Univ. of Toronto, preprint, (1994).
11. Oja, E., "Neural networks, principal components, and subspaces", *Int. J. Neural Systems* 1, 61-68 (1989).
12. Xu, L., "Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach: (II) Supervised Learning", Invited paper, to appear on *Proc. of Intl Workshop on Theoretical Aspects of Neural Computation*, May 26-28, Hong Kong, Springer-Verlag, (1997).
13. Xu, L., "Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach: (III) Time Series Learning", to be submitted, (1997).
14. Xu, L., "New Advances on Bayesian Ying-Yang Learning System With Kullback and Non-Kullback Separation Functionals", *Proc. of 1997 IEEE Intl. Conf on Neural Networks (IEEE-INNS IJCNN97)*, June 9-12, Houston, TX, USA, Vol. III, pp1942-1947(1997).
15. Xu, L., "Bayesian Ying-Yang Learning: From Multilayer Nets To Mixture of Experts and Radial Basis Function Nets ", To appear on *Algorithmica*, a special issue on Computational Learning Theory, (1997).
16. Xu, L., "Bayesian Ying-Yang Machine, Clustering and Number of Clusters", to appear on *pattern Recognition Letters*, (1997).
17. Xu, L., "Bayesian Ying-Yang Learning Based ICA Models" , to appear on Proc. 1997 IEEE Workshop on Neural Networks and Signal Processing, 24-26 Sept., Florida, (1997).
18. Xu, L., Cheung, C.C. , Yang, H.H., . and Amari , S., " Independent component analysis by the information-theoretic approach with Mixture of Density ", Invited special session on Blind Signal Separation, *Proc. of 1997 IEEE Intl. Conf on Neural Networks (IEEE-INNS IJCNN97)*, June 9-12, Houston, TX, USA, Vol. III, pp1821-1826(1997).
19. Xu, L., Cheung, C.C. , Ruan, J. and Amari , S., "Nonlinearity and Separation Capability: Further Justification for the ICA Algorithm with A Learned Mixture of Parametric Densities, Proc. of 1997 European Symp. on Artificial Neural Networks, Bruges, April 16-18, pp291-296(1997).
20. Xu, L., "Bayesian-Kullback YING-YANG Learning Scheme: Reviews and New Results", *Proc. Intl Conf. on Neural Information Processing (ICONIP96)*, Sept. 24-27, Springer-Verlag, pp59-67(1996).
21. Xu, L., "Bayesian-Kullback YING-YANG Machines for Supervised Learning", Invited Talk, *Proc. of 1996 World Congress On Neural Networks*, Sept. 15-18, San Diego, CA, pp193-200(1996).
22. Xu, L. "A Unified Learning Scheme: Bayesian-Kullback YING-YANG Machine", *Advances in Neural Information Processing Systems 8*, eds., David S. Touretzky, Michael Mozer, Michael Hasselmo, MIT Press, Cambridge MA, pp444-450(1996).
23. Xu, L. , "How Many Clusters ? : A YING-YANG Machine Based Theory For A Classical Open Problem In Pattern Recognition", Invited Talk, *Proc. of 1996 IEEE Intl. Conf. on Neural Networks*, Washington, DC, June 2-6, Vol.3, 1546-1551(1996).
24. Xu, L., & Jordan, M. I. , " On convergence properties of the EM algorithm for Gaussian mixtures" , *Neural Computation 8*, (2), 129–151(1996).
25. Xu, L., "YING-YANG Machine: a Bayesian-Kullback scheme for unified learnings and new results on vector quantization", Keynote talk, *Proc. Intl Conf. on Neural Information Processing (ICONIP95)*, Oct 30 - Nov. 3, pp977-988(1995).
26. Xu, L., "YING-YANG Machine for Temporal Signals", Keynote talk, *Proc IEEE Intl. Conf. Neural Networks & Signal Processing*, Nanjing, Dec.10-13, Vol. I, pp644-651(1995).
27. Xu, L., "New Advances on The YING-YANG Machine", Invited paper, *Proc. of 1995 Intl. Symp. on Artificial Neural Networks*, Dec. 18-20, Hsinchu, Taiwan, ppIS07-12(1995).
28. Xu, L., " A Unified Learning Framework: Multisets Modeling Learning", Invited Talk, *Proc. of World Congress On Neural Networks*, July 17-21, Washington, DC, Vol.I, pp35-42(1995).

29. Xu, L. , "Cluster Number Selection, Adaptive EM Algorithms and Competitive Learnings", Invited Talk, *Proc IEEE Intl. Conf. Neural Networks & Signal Processing*, Nanjing, Dec.10-13, Vol.II, pp1499-1502(1995).
30. Xu, L. , "Multisets Modeling Learning: An Unified Theory for Supervised and Unsupervised Learning", Invited Talk, *Proc. of 1994 IEEE Intl. Conf. on Neural Networks*, June 26-July 2, Orlando, FL, Vol.I, pp315-320(1994).
31. Xu, L. , " Beyond PCA Learnings: From Linear to Nonlinear and From Global Representation to Local Representation ", Invited Talk, *Proc. of 1994 Intl. Conf. on Neural Information Processing(ICONIP94)*, Oct 17-20, Seuol, Korea, pp943-949(1994).
32. Xu, L., " Theories for Unsupervised Learning: PCA and Its Nonlinear Extensions", Invited Talk, *Proc. of 1994 IEEE Intl. Conf. on Neural Networks*, June 26-July 2, Orlando, FL, pp1252-1257(1994).
33. Xu, L., Krzyzak, A. and Oja, E., " Rival Penalized Competitive Learning for Clustering Analysis, RBF net and Curve Detection", *IEEE Trans. on Neural Networks 4*, No.4, pp636-649(1993).
34. Xu, L, "Least mean square error reconstruction for self-organizing neural-nets", *Neural Networks 6*, pp. 627-648(1993.) (Its early version on *Proc. of 1991 Intl. Joint Conf. on Neural Networks (IJCNN91'Singapore*), 2363-2373, (1991).
35. Xu, L, Oja, E. & Suen, C. Y., "Modified Hebbian learning for curve and surface fitting"., *Neural Networks 5*, 441-457(1992).

Acknowledgement. This work was supported by the HK RGC Earmarked Grants CUHK250/94E and CUHK484/95E and by Ho Sin-Hang Education Endowment Fund for Project HSH 95/02.

The basic ideas of the BYY learning system and theory in this paper and its sister paper [12] as well as several 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. This paper and its sister paper are both completed in the first week that HK, a harmony of the eastern and western cultures, returned to China and thus I myself 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.